МАТЕМАТИЧКИ ФАКУЛТЕТ
УНИВЕРЗИТЕТ У БЕОГРАДУ

Мастер рад

Комбинација бинарног уређеног стабла и хипа

ментор:
проф. др Миодраг Живковић

студент:
Јовица Топалоски

комисија:
проф. др Предраг Јаничић
проф. др Саша Малков

Септембар, 2016.

Садржај

Сажетак

У овом раду описан је трип, структура података која представља комбинацију бинарног уређеног стабла и хипа. Описане су операције које омогућава ова структура: тражење кључа, уметање, брисање, подела структуре на две мање и спајање у већу структуру. Поред тога, описано је рандомизирано стабло претраге као најважнија структура података која се заснива на трипу.

У првом поглављу описан је трип као структура података. Друго поглавље садржи основне појмове о трипу и о могућим операцијама са трипом: тражење датог кључа, умета-ње новог кључа, брисање кључа, подела трипа на два мања и спајање два трипа у један. У поглављу три описано је рандомизирано стабло претраге, које представља специјалну примену трипа. У поглављу четири описана је анализа рандомизираног стабла. У поглављу пет анализиране су операције рандомизираног стабла. У поглављу шест описана је програмска реализација трипа и спроведени експерименти. У поглављу седам је кратак закључај рада.

Већи део рада, тачније поглавља о рандомизираном стаблу претраге и анализа рандомизираног стабла и операција настала је на основу [1], а коришћене су и лекције са универзитетских курсева [2, 3]. У програмској имплементацији искоришћена је функција *randomVal* преузета са сајта [4]. Функција *printTreap* се зансива на делу кода са сајта [5]. За експеримент мерења времена претраге речи у речнику коришћен је помоћни програм чији се један део заснива на коду са сајта [6].

Поглавље 1

Увод

Све присутан проблем у рачунарској науци представља чување скупа кључева и брз приступ истим. Када треба чувати кључ из неког уређеног скупа обично се користи нека врста стабла претраге. У бинарном стаблу претраге кључ сваког чвора већи је од кључева свих чворова левог подстабла, а мањи од кључева чворова десног подстабла. Оно представља најједноставнији пример стабла претраге. Време потребно за приступање неком податку у чвору суштински је одређено максималном дубином чвора у стаблу у којем се податак налази. Зато је пожељно да сваки чвор има што мању дубину.

Како се скуп чворова временом мења и како се чворови могу убацивати и брисати непредвидиво, не може се очекивати да се аутоматски обезбеди мала максимала дубине чворова. Могуће решење проблема је коришћење специјалних варијанти бинарних стабала претраге, које гарантују уравнотеженост, односно да све дубине чворова буду $O(\log_2 n)$(у наставку текста ће се подразумевати да је основа 2), где је n број елемената у стаблу. Типично је да овакве класе стабла омогућују операције приступа чвору и ажурирања стабла у времену $O(\log n)$ у најгорем случају и могу се имплементирати тако да се ове измене раде помоћу ротација за уравнотежавање стабла. Ротације ће бити објашњене у следећем поглављу. Ако се неком чвору приступа чешће него осталима згодно је такве чворове стављати ближе корену уколико је то могуће.

Хип је бинарно стабло које задовољава услов хипа: кључ сваког чвора већи је или једнак од кључева његових синова. Трип, комбинација бинарног стабла претраге и хипа представља структуру података у којој сваки чвор има кључ за претрагу и приоритет. Кључеви чворова задовољавају исти услов као у бинарном стаблу претраге. С друге стране

приоритети чворова задовољавају услов да је приоритет оца већи од приоритета сина. Ова структура је још једно ефикасно решење структуре података за скупове кључева. Као и код хипа, последица је да је корен чвор са највећим приоритетом. Чворови чији кључеви су мањи од кључа у корену се налазе у левом подстаблу, односно они са већим у десном. Приоритети се бирају случајно при упису кључа.

Ову структуру су 1989. предложили **С. Арагон**(C. Aragon) и **Р. Сеидел**(R. Seidel) и на основу ње изградили рандомизирано стабло претраге. Они су такође предложили и начин за оптимизацију структуре.

Поглавље 2

Трип и операције

У овом поглављу објашњен је појам трипа, описане су основне операције као што су: тражење елемента са задатим кључем, уметање новог кључа, брисање кључа, подела трипа на два и спајање два трипа са раздвојеним величинама кључева у један. Објашњене су ротације које се користе како би се очували услови бинарног стабла и хипа у трипу.

2.1 Основни појмови

Дефиниција 1 *Нека је X скуп ставки од којих свака има придружен кључ и приоритет, где се кључеви и приоритети бирају из потпуно уређених скупова. Потребно је да овискупови буду различити. Трип за скуп X може се формирати као уређено стабло за кључеве и хип за приоритете. Чворови чији су кључеви мањи од кључа корена налазе се у левом а кључеви чији су чворови већи од кључа корена у десном подстаблу. Приоритет оца је увек већи од приоритета сина, чиме је задовољен услов хипа.*

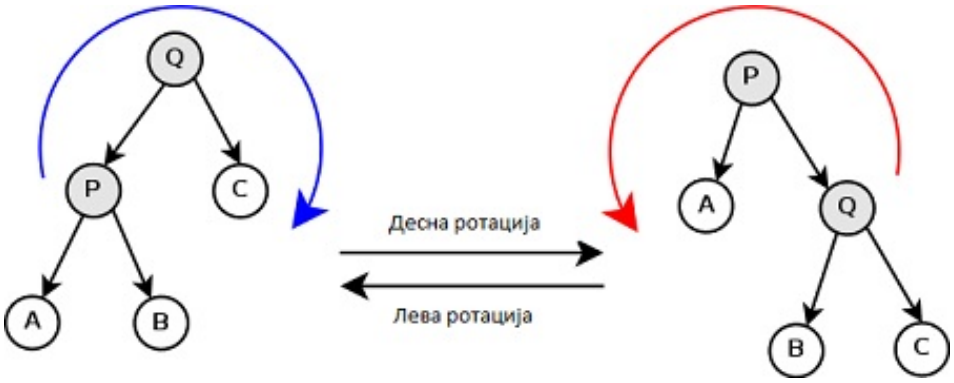
Теорема 1 *За сваки скуп X постоји трип.*

Доказ. Под претпоставком да су сви кључеви и приоритети различити, разумна претпоставка би била да је корен чвор са највећим приоритетом, а распоред осталих чворова лево и десно одређен је по њиховим кључевима. Трип за скуп чворова X представља бинарно стабло претраге које је резултат сукцесивног уметања чворова из X у опадајућем редоследу приоритета у иницијално празно стабло користећи алгоритам за уметање у лист. ■

Ротација стабла је операција над бинарним уређеним стаблом која мења структуру тако што један чвор (оца) спушта један ниво наниже, а други (сина) подиже један ниво навише заједно са одговарајућим подстаблима. Код леве ротације син Q постаје нови корен подстабла, а отац P постаје леви син и наслеђује десног сина. Код десне ротације син P постаје отац подстабла, а отац Q постаје син P и наслеђује десног сина. Ротације су приказане на слици 2.1. У наставку је дат псеудо код процедура ових ротација. Структура чвора садржи показиваче на левог и десног сина (*lchild*, *rchild*), при чему је T показивач на чвор (корен трипа).

```
treap* RightRotation(treap* T){
    treap* lT = T->lChild;
    T->lChild = lT->rChild;
    lT->rChild = T;
    return lT;
}


treap* LeftRotation(treap* T){
    treap* lT = T->rChild;
    T->rChild = lT->lChild;
    lT->lChild = T;
    return lT;
}
```



Слика 2.1 *Операције ротације*

У наставку је дат кратак опис основних операција тражења, уметања и брисања чвора трипа. Све операције са трипом биће детаљније објашњене касније у тексту. Нека је T трип за чување чворова из скупа X. Дат је кључ $x \in X$ који се може лако наћи у T помоћу алгоритма претраге. Време неопходно за овакав приступ пропорционално је дубини T. Уметање нове ставке z у T врши се тако што се z уметне у одговарајући лист у T. Како може доћи до нарушавања услова хипа, могуће је да ће бити потребне ротације приказане на слици 2.1. Брисање се може извести инверзним редоследом у односу на уметање.

Трип се може поделити на два мања трипа, један који садржи кључеве мање од задатог броја x, и други који садржи веће. Чвор са кључем x потребно је убацити у задати трип тако да задовољава услов да је његов приоритет већи од свих осталих приоритета. Након уметања он постаје корен трипа, сви кључеви мањи од x ће се налазити у његовом левом подстаблу, а већи у десном.

За спајање два трипа (довољно је да су им кључеви раздвојени по величини - сви кључеви једног су мањи од кључева другог), у један креира се трип са кључем x као горе који је корен, чије је лево подстабло $T1$, а десно $T2$ и примени се операција брисања чвора x.

Познавање показивача на чвор у трипу може да убрза неке операције. На пример, ако се зна показивач на чвор x, онда се време брисања тог чвора смањује на време ротација чвора x у лист и брисање листа, без потребе за претрагом. Сваки чвор сем корена повезан је са само једним чвором на претходном нивоу. Следбеник чвора x је чвор који има најмањи кључ $y > x$. Слично и уметање чвора x може бити убрзано уколико се зна показивач на његовог претходника (или следбеника) s. Приликом уметања кључа креира се нови чвор (лист) у који се смешта кључ. Претрага за одговарајућом позицијом на којој ће бити прикачен лист, креће од чвора s уместо од самог корена. Такође операције раздвајања и спајања трипа могу бити убрзане уколико се знају показивачи на чворове са минималним и максималним кључевима у оба хипа (опширније о операцијама раздвајања и спајања речено је у поглављима 5.7 и 5.8).

2.2 Тражење елемента са задатим кључем

Тражење датог кључа примењује се стандардни алгоритам претраге бинарног стабла, при чему се игноришу приоритети у трипу. Потребно

је да се пронађе елемент у структури чији кључ је једнак задатом или установити да га нема. Задати кључ x се прво пореди са кључем корена p. Ако је $x{=}p$ претрага је завршена. У случају $x{<}p$ покреће се рекурзивна претрага левог подстабла, односно десног за $x{>}p$. Структура чвора трипа садржи кључ, приоритет и показиваче на ледо и десно подстабло.

Algoritam nadji_ u_ BSP (Koren,x)
Ulaz: *Koren (pokazivač na koren BSP), x(broj).*
Izlaz: *Pokazivač na čvor koji sadrži ključ x, ili nil ako takav ne postoji.*

begin
  if Koren = **nil or** Koren.Ključ = x **then** Čvor ← Koren
  else
    if $x <$ Koren.Ključ **then** Čvor ← nadji_u_BSP(Koren.levi, x)
    else Čvor ← nadji_u_BSP(Koren.desni, x)
  return Čvor
end

Сложеност ове операције зависи од облика стабла и положаја чвора на коме се врши интервенција. У најгорем случају овај алгоритам мора да тражи од корена стабла до листа који му је најудаљенији. Све остале операције захтевају константан број корака. Уколико се претрага завршава у листу, резултат је да је сложеност пропорционална висини стабла тј. $O(\log n)$(видети [4]), где је n број чворова.

2.3 Уметање новог кључа

Уметање новог кључа подразумева убацивање задатог кључа у трип на одређеној позицији. На којој позицији ће кључ бити уметнут зависи од приоритета који се додељује кључу. Проблеми који могу да настану при операцији уметања везани су за уравнотеженост стабла. Ови проблеми се решавају операцијом ротације која је објашњена у претходном делу текста и илустрована на слици 2.1.

За уметање новог кључа x у трип T, најпре је потребно генерисати случајни приоритет y за x. Потребно је креирати нови чвор у који се смешта кључ x. Чвор треба ставити у лист на одговарајућој позицији, која се одређује алгоритмом за бинарну претрагу за кључ x. Чвор се креира на месту где се претрага завршила. Када се убаци нови чвор, кључеви у модификованом стаблу се налазе у уређеном поретку, међутим

може се десити да услов хипа није задовољен. Стога се примењују ротације све док је приоритет чвора x већи од приоритета оца или док не буде у корену. Резултат је да је сложеност ове операције $O(\log n)$(видети [4]), где је n број чворова. Процедура уметања у трип дата је следећим псеудо кодом у ком класа $Treap$ има поља за чување кључа. приоритета и показиваче на лево и десно подстабло:

Algoritam Upis_ u_ trip $(x.key, x.priority, Treap\ T)$
if T = **null then** Kreirati novi čvor T
gde je T.key jednako x.key, T.priority jednako x.priority,
T.lchild je null, and T.rchild je null
else if x.key<T.key **then** Upis_u_trip$(x.key, x.priority, T.lchild)$
if T.lchild.priority>T.priority **then** DesnaRotacija(T)
else if x.key>T.key **onda** Upis_u_trip$(x.key, x.priority, T.rchild)$
if T.rchild.priority>T.priority **then** LevaRotacija(T)
else x je već u T

2.4 Брисање кључа

Брисање кључа из трипа не сме да наруши сортирани редослед кључева и услов хипа за приоритете. То се постиже тако што се бришу чворови који су листови. Ако је чвор x који се брише лист, онда се брише показивач на њега у његовом оцу. Ако x није лист онда се низом ротација чвор спушта док не буде у листу. Низ примењених ротација зависи од приоритета. Ако је леви син чвор са мањим приоритетом примењује се десна ротација, а у супротном лева. Ротације могу да изазову нарушавање услова хипа, па су у том случају потребне додатне ротације. Резултат је да је сложеност операције брисања кључа једнака $O(\log n)$(видети [4]), где је n број чворова. Следећи псеудо код описује процедуру брисања из трипа:

Algoritam Brisanje_ čvora $(x : key, T : treap)$
nadji_u_BSP(x, T)
while x nije list
u← sin sa manjim prioritetom
if u levi sin **then** DesnaRotacija(x)
else LevaRotacija(x)
Obriši x

Пример 2.4.1 *На слици 2.2 приказан је пример брисања и уметања чвора у трип. Чвор са кључем L и приоритетом* 69 *се брише(умеће) у трип. У примеру брисања види се како се чвор низом ротација спушта до листа одакле се брише. У случају уметања чвора, прво се умеће у лист, а затим се низом ротација чвор пење док не дође на одговарајуће место тако да задовољи услов хипа.*

2.5 Подела трипа на два мања

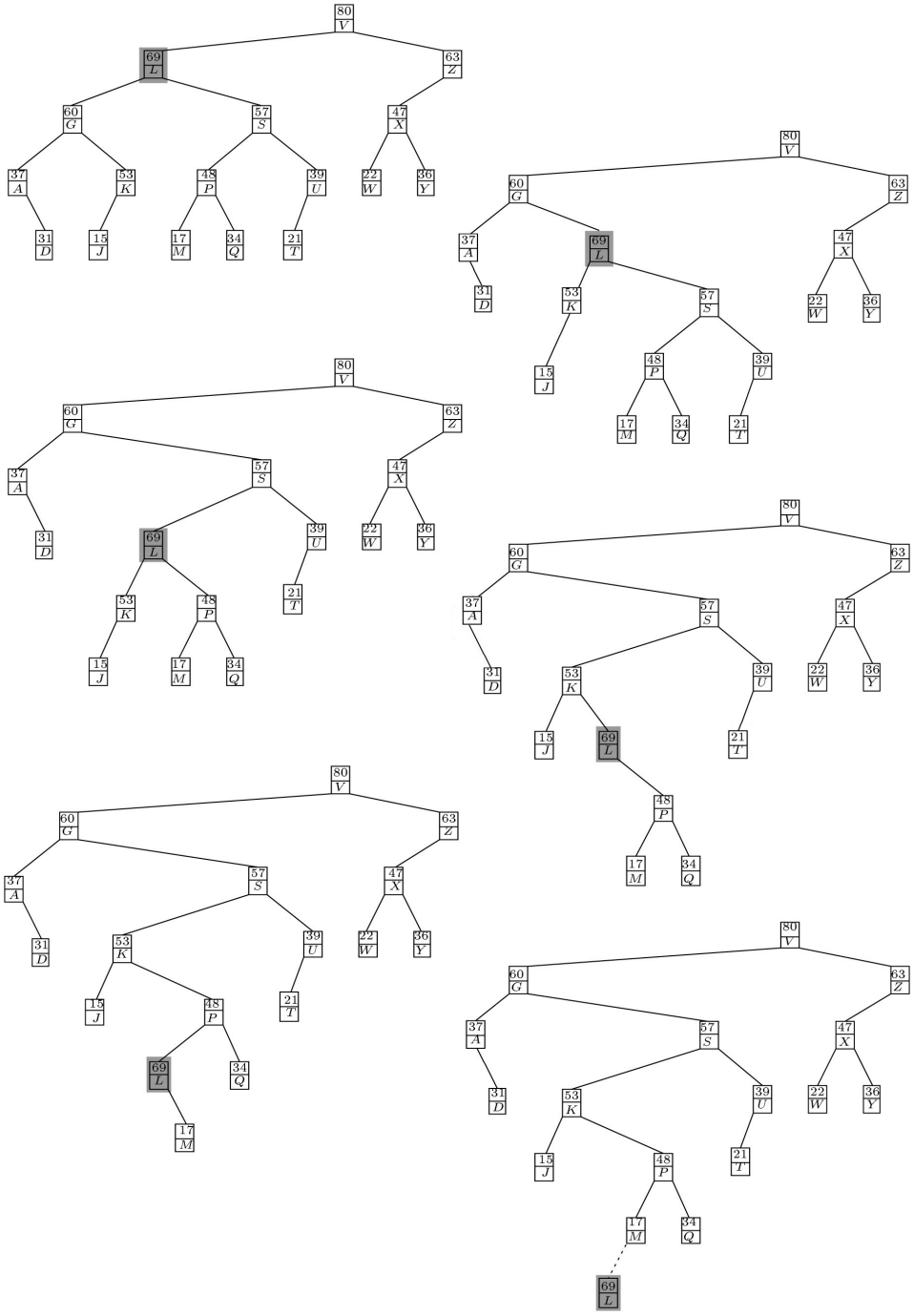
Нека је задат трип T са скупом кључева X, и нека је задат кључ $a \in X$. Задатак је поделити T на два мања трипа $T1$ и $T2$ тако да $T1$ садржи кључеве из $X1 = \{x \in X | x.key \leq a\}$, а $T2$ кључеве из $X2 = \{x \in X | x.key > a\}$. Да би се трип поделио на два према кључу a, треба тај кључ убацити у трип тако да постане корен трипа. Приоритет таквог чвора a мора бити највећи приоритет у трипу. Лево подстабло представља трип $T1$, а десно $T2$. Резултат је да је сложеност операције поделе трипа на два мања једнака $O(\log n + \log m)$(видети [4]), где је n број чворова једног трипа,а m број чворова другог трипа. Следи псеудо код процедуре за поделу трипа:

Procedrura PodelaTripa(T: treap, a:key, T1, T2: treap)
    Upis_u_trip((a,∞), T)
    T1 $\leftarrow$ T $\rightarrow$ lchild
    T2 $\leftarrow$ T $\rightarrow$ rchild

Пример 2.5.1 *На слици 2.3 приказан је пример поделе трипа на два мања. Кључеви чворова су карактери са лексикографским уређењем. Прво се у трип који треба поделити додаје чвор т са приоритетом који је већи од свих осталих приоритета у трипу, тако да он постане корен трипа. Леви и десни син су чворови са кључевима с и р који представљају корене два жељена трипа након операције поделе.*

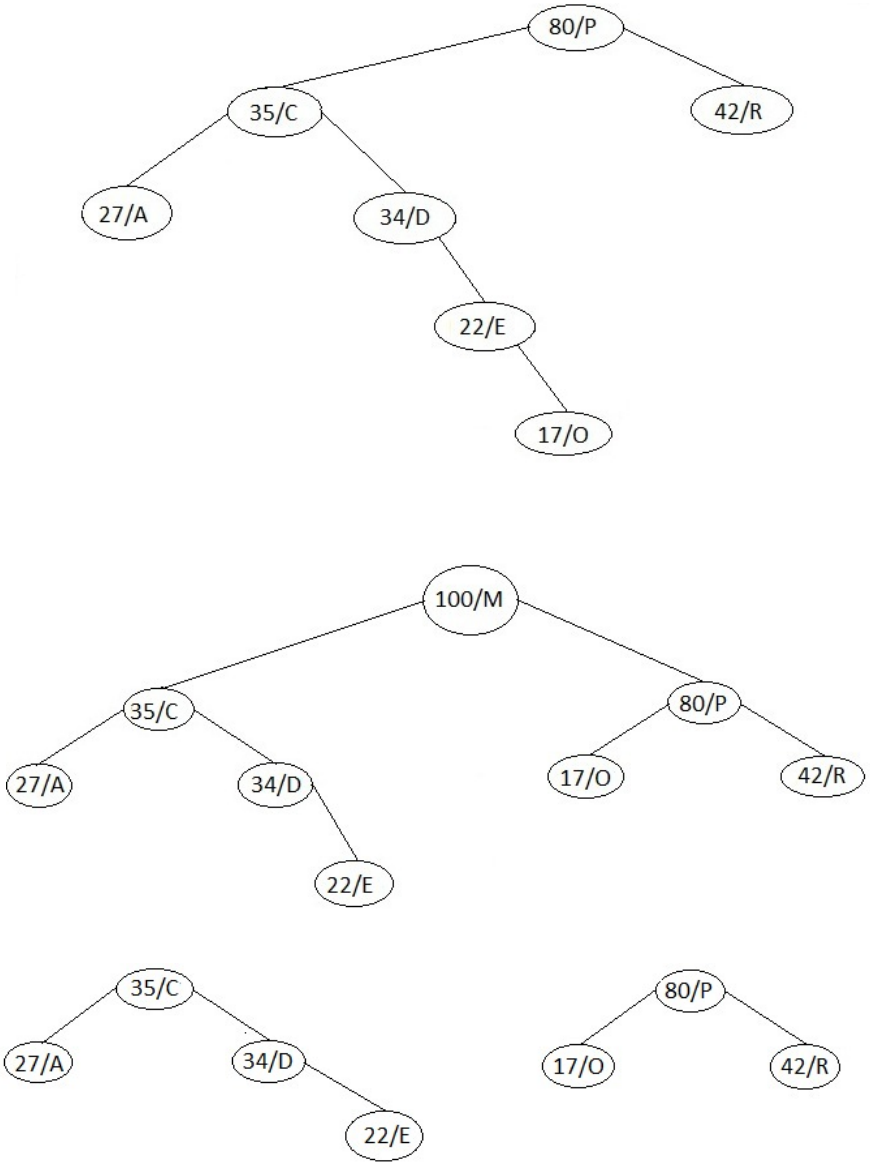
2.6 Спајање два дата трипа у један

При операцији спајања два трипа $T1$ и $T2$ у један, подразумева се да је највећи кључ у трипу $T1$ мањи од најмањег у $T2$. Креира се нови чвор са кључем x такав да је кључ x већи од максималног кључа трипа $T1$, а мањи од минималног кључа у $T2$ и додељује му се најмањи



Слика 2.2 *Брисање/уметање чвора*

приоритет. Затим се за левог сина постави корен трипа $T1$, а за десног корен трипа $T2$. Сада су трип $T1$ и $T2$ лево и десно подстабло чвора



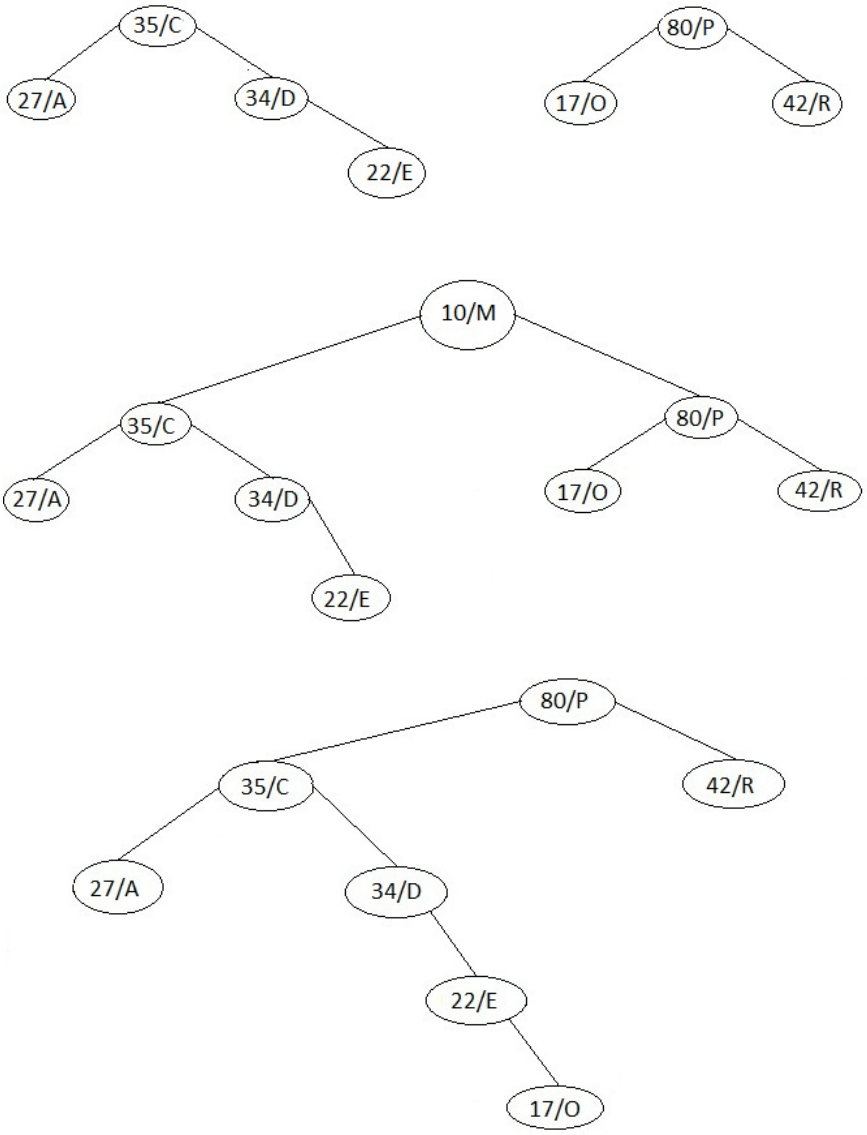
Слика 2.3 *Пример поделе трипа на два мања*

са кључем x. Како x има најмањи приоритет потребно је урадити пар ротација да би био задовољен услов хипа. Чвор x се спушта у лист одакле се касније брише помоћу већ описане операције брисања из трипа. Као резултат добија се један трип састављен од два оригинална. Ова операција представља инверзни поступак операције поделе већег трипа на два мања. Резултат је да је сложеност операције поделе трипа на два мања једнака $O(\log n + \log m)$(видети [4]), где је n број чворова једног

трипа,а m број чворова другог трипа. Дат је псеудо код процедуре ове операције:

Procedrura SpajanjeTripa(T1, T2, T: treap)
$T \leftarrow$ Kreiraj_novi_čvor(x)
$T \rightarrow$ [lchild] $\leftarrow [T1]$
$T \rightarrow$ [rchild] $\leftarrow [T2]$
Brisanje_čvora(x)

Пример 2.6.1 *На слици 2.4 приказан је пример спајања два трипа добијена операцијом поделе у један. Кључеви чворова су карактери са лексикографским уређењем. Прво се креира чвор са кључем (m) који је већи од кључа e, а мањи од o и додели му се најмањи приоритет. После уписа тог чвора у трип, његов леви, односно десни син постају чворови са кључем c, односно p. Низом ротација чвор m се спушта у лист одакле се брише, а као резултат се добије један трип.*



Слика 2.4 *Пример спајања два трипа добијена операцијом поделе*

Поглавље 3

Рандомизирано стабло претраге

Нека је X скуп чворова од којих је сваки једнозначно одређен кључем из потпуно уређеног скупа. Рандомизирано стабло претраге за скуп X дефинише се као трип за X, где су приоритети чворова независне непрекидне случајне променљиве. Наредну теорему наводимо без доказа, видети [1]:

Теорема 2 *Рандомизирано стабло претраге које има n чворова има перформансе које су приказане у табели 3.1.*

Нека је F вероватноћа расподеле. Нека је X скуп чворова означених кључевима из уређеног скупа, при чему је сваком $x \in X$ придружена целобројна тежина $w(x) > 0$. Уравнотежено рандомизирано стабло за X се дефинише као трип за X, при чему су приоритети независне непрекидне случајне променљиве дефинисане на следећи начин: приоритет елемента x је максимум $w(x)$ независних случајних променљивих са вероватноћом F.

Теорема 3 *Нека је T уравнотежено рандомизирано стабло претраге за скуп чворова X са тежинама. У табели 3.2 су приказане очекиване перформансе рандомизираног стабла T. Овде W означава суму тежина чворова из X. За чвор x претходник и следбеник у стаблу су означени са x^- и x^+. $Tmax$ и $Tmin$ означавају чвор са минималним и максималним кључем.*

Мерило перформанси	Очекиване границе
време приступа	$O(\log n)$
време уметања	$O(\log n)$
време уметања за елемент са ручком(на претх. или следб.)	$O(1)$
време брисања	$O(\log n)$
време брисања елемента са ручком	$O(1)$
број ротација по ажурирању	2
време претраге са показивачем преко дистанце d	$O(\log d)$
време брзе претраге са показивачем преко дистанце d	$O(\log min\{d, n-d\})$
време спајања два трипа величине m и n	$O(\log max\{m, n\})$
време поделе трипа на два мања величине m и n	$O(\log max\{m, n\})$
време брзе поделе или спајања	$O(\log min\{m, n\})$
време одсецања стабла величине d	$O(\log n)$
време ажурирања ако ротација подстабла величине s кошта $O(s)$	$O(\log n)$
време ажурирања ако ротација подстабла величине s кошта $O(s \log^k s), k \geq 1$	$O(\log^{k+1} n)$
време ажурирања ако ротација подстабла величине s кошта $O(s^a)$, a$>$1	$O(n^{a-1})$
време ажурирања ако ротација кошта $f(s)$, $f(s)$ не-негативна	$O(\frac{f(n)}{n} + \Sigma_{0<i<n} \frac{f(i)}{i^2})$

Табела 3.1: Времена извршавања операција рандомизараног стабла претраге

Запази се пре свега да нема претпоставки око расподеле кључева у X. Претпоставимо за сада, да су приоритети скривени од корисника. Ово је неопходно да би се рандомизирано стабло претраге трансформисало при сваком ажурирању у рандомизирано стабло. Ако би корисник знао приоритете могао би да креира не рандомизирано и небалансирано стабло

Мерило перформанси	Очекиване границе
време приступа чвору x	$O(1+\log\frac{W}{w(x)})$
време уметања чвора x	$O(1+\log\frac{W+w(x)}{min\{w(x^-),w(x),w(x^+)\}})$
време брисања чвора x	$O(1+\log\frac{W}{min\{w(x^-),w(x),w(x^+)\}})$
време уметања чвора x са ручком на претходнику	$O(1+\log(1+\frac{w(x)}{w(x^-)}+\frac{w(x)}{w(x^+)}+\frac{w(x^-)}{w(x^+)}))$
време брисања чвора x са ручком	$O(1+\log(1+\frac{w(x)}{w(x^-)}+\frac{w(x)}{w(x^+)}))$
број ротација по ажурирању чвора x	$O(1+\log(1+\frac{w(x)}{w(x^-)}+\frac{w(x)}{w(x^+)}))$
време потребно за "finger" претрагу између чвора x и y, где је V укупна тежина чворова између x и y(укључујући и њих)	$O(\log\frac{V}{min\{w(x),w(y)\}})$
време потребно за спајање два стабла $T1$ и $T2$ тежина $W1$ и $W2$	$O(1+\log\frac{W1}{w(T1max)}+\log\frac{W2}{w(T2min)})$
време потребно за поделу трипа T на два мања $T1$ и $T2$, тежина $W1$ и $W2$	$O(1+\log\frac{W1}{w(T1max)}+\log\frac{W2}{w(T2min)})$
време потребно за брзо спајање или брзу поделу	$O(1+\log min\{\frac{W1}{w(T1max)},\frac{W2}{w(T2min)}\})$
време потребно за увећање тежине чвора x за Δ	$O(1+\log\frac{w(x)+\Delta}{w(x)})$
време потребно за умањење тежине чвора x за Δ	$O(1+\log\frac{w(x)}{w(x)-\Delta})$

Табела 3.2: Времена извршавања операција тежинског рандомизараног стабла претраге

после полиномијално много ажурирања.

Захтев да се као приоритети користе непрекидне случајне променљиве није неопходан. То се ради како би се са вероватноћом 1 добили различити приоритети. Чак и ако се пређе на дискретну расподелу, вероватноћа да има више њих истих је јако мала. Ово значи да је у пракси за већину апликација довољно користити бројеве из скупа целих бројева као на пример $[0, 2^{31}]$. У поређењу са другим балансираним стаблима која захтевају по 1 бит информација по чвору, рандомизирано стабло претраге захтева више информација, што може бити разочаравајуће. Начини како се ово може избећи описује се у поглављу пет.

Сваки пут када се приступа чвору x генерише се нови случајни број r према расподели F. Ако је број r већи од приоритета чвора x, онда он треба да постане његов приоритет и ако је потребно урадити ротације да се чвор x попне навише у стаблу како не би био нарушен услов хипа. Ако се x-у приступа k пута, његов приоритет ће бити максимум k случајних бројева. Последица те чињенице, која се без доказа наводи у [1] је да је очекивана дубина чвора $O(\log(1/p))$, где је $p = k/A$, при чему је A број приступа у целом стаблу.

Поставља се питање како уметнути чвор x у уравнотежено рандомизирано стабло са фиксном тежином k. Ово се може најлакше урадити ако се за функцију расподеле узме функција равномерне расподеле вероватноће на интервалу [0,1]. Функција расподеле G_k за k случајних променљивих има форму $G_k(z) = z^k$. Из овога следи да x треба да се уметне са приоритетом $r^{1/k}$ где је r случајан број из [0,1]. Како је поређење једина операција која укључује приоритете, а логаритамска функција је монотона, онда се x може уметнути са приоритетом $(\log r)/k$ уместо $r^{1/k}$.

Поставља се питање колико случајних битова је потребно да би се имплементирало рандомизирано стабло претраге. Постоји једноставна стратегија која показује да је потребан исти број случајних битова по ажурирању у неуравнотеженом стаблу. Ово се може постићи на следећи начин: нека су приоритети реални случајни бројеви из интервала [0,1]. Такви бројеви могу бити генерисани додавањем случајних битова за проши-рење бинарне репрезентације на њихову бинарну репрезентацију када је то потребно због поклапања приоритета два чвора. Испоставља се да се генерише само онолико бинарне репрезентације колико је потребно. Једина операција са приоритетима у алгоритмима ажурирања је поређење приоритета. У многим случајевима резултат таквог поређења ће већ бити одређен постојећом бинарном репрезентацијом. Када ово није случај, једна репрезентација постаје префикс друге. Испоставља се да је очекивани број додавања битова потребних за решавања поређења не већи од четири. Пошто се у алгоритмима ажурирања приоритети пореде само у вези са ротацијом, а број очекиваних ротација по ажурирању је мањи од два, значи да је за уметање број очекиваних случајних битова мањи од дванаест, а за брисање мањи од осам.

У раду [1] показано је да оцене сложености операција са рандомизираним стаблом настављају да важе и ако се примењују приоритети са само $O(\log n)$ случајних битова.

Поглавље 4

Анализа рандомизираног стабла претраге

У овом одељку дата је анализа величина рандомизираног стабла претраге и извођење њихових очекиваних вредности.

Дефиниција 2 *Дужина леве(десне) кичме је дужина пута од корена када се силази само преко левих(десних) синова.*

При томе се користе следеће ознаке:
$D(x)$ - дубина чвора x у стаблу (број чворова на путу од корена до чвора x),
$S(x)$ - величина подстабла са кореном x (број чворова у подстаблу),
$P(x, y)$ - дужина јединствене путање од чвора x до чвора y,
$SL(x)$ и $SR(x)$ - дужина десне кичме левог подстабла и дужина леве кичме десног подстабла, за леву страну се подразумева лева страна заједно са чвором корена, десна страна је дефинисана аналогно.

У овом делу биће речи о трипу T за скуп $X = \{x_i = (k_i, p_i) | 1 \leq i \leq n\}$ од n чворова, нумерисаних кључевима у растућем поретку $k_i < k_{i+1}$ за свако $1 \leq i < n$. За приоритет p_i узимају се случајне променљиве. Како за фиксни скуп кључева трип T зависи од приоритета, горе наведен величине ће такође бити случајне променљиве, за које има смисла анализирати очекивања, вероватноће и слично.

Анализа је поједностављена чињеницом да се свака горе наведена величина може представити помоћу два типа индикатора случајних променљивих:
$A_{i,j}$ је 1 ако је x_i предак x_j у T и 0 иначе,

$C_{i;l,m}$ је 1 ако је x_i заједнички предак x_l и x_m у T и 0 иначе.
Уз то подразумева се да је сваки чвор предак самом себи.

Теорема 4 *Нека је* $1 \leq l, m \leq n$ *и нека је* $l < m$. *Тада је*
(I) $D(x_l) = \Sigma_{1 \leq i \leq n} A_{i,l}$
(II) $S(x_l) = \Sigma_{1 \leq j \leq n} A_{l,j}$
(III) $P(x_l, x_m) = 1 + \Sigma_{1 \leq i < m}(A_{i,l} - C_{i;l,m}) + \Sigma_{l < i \leq n}(A_{i,m} - C_{i;l,m})$
(IV) $SL(x_l) = \Sigma_{1 \leq i < l}(A_{i,l-1} - C_{i;l-1,l})$; $SR(x_l) = \Sigma_{l < i \leq n}(A_{i,l+1} - C_{i;l,l+1})$

Доказ. Ставке (I) и (II) су јасне, јер је дубина чвора ништа друго него број предака, и величина његовог подстабла је број чворова за коју је предак.
За ставку (III) треба напоменути да је пут од x_l до x_m подељен на два дела најмањим заједничким претком v. Део од (x_l) до v садржи управо претке од (x_l) без заједничких предака од (x_l) и (x_m), чији је број једнак првој суми. Слично се рачуна и друга сума за пут од (x_m) до v. Како се v не броји у оба дела потребна је корекција од $+1$.

За ставку (IV) довољно је размотрити случај SL. Ако (x_l) има лево подстабло онда најмањи чвор у десном делу мора бити (x_{l-1}). Јасно је да у том случају десна страна садржи тачно претке (x_{l-1}) без предака (x_l). Такође (x_i) са $i > l$ може бити на десној страни. Ако (x_l) нема лево подстабло или је $l = 1$ у том случају формула има тачно вредност 0 или је (x_{l-1}) предак од (x_l), у ком случају је сваки предак од (x_{l-1}) заједнички предак од (x_l) и (x_{l-1}) и формула има тачно вредност 0. ■

Ако узмемо да је $a_{i,j} = Ex[A_{i,j}]$ и $c_{i;l,m} = Ex[C_{i;l,m}]$ због линеарности математичког очекивања следи:

Последица 1 *Нека је* $1 \leq l, m \leq n$ *и* $l < m$,
(I) $Ex[D(x_l)] = \Sigma_{1 \leq i \leq n} a_{i,l}$
(II) $Ex[S(x_l)] = \Sigma_{1 \leq j \leq n} a_{l,j}$
(III) $Ex[P(x_l, x_m)] = 1 + \Sigma_{1 \leq i < m}(a_{i,l} - c_{i;l,m}) + \Sigma_{l < i \leq n}(a_{i,m} - c_{i;l,m})$
(IV) $Ex[SL(x_l)] = \Sigma_{1 \leq i < l}(a_{i,l-1} - c_{i;l-1,l})$; $Ex[SR(x_l)] = \Sigma_{l < i < n}(a_{i,l+1} - c_{i;l,l+1})$

У суштини цела прича о анализи своди се на очекивања $a_{i,j}$ и $c_{i;l,m}$. Како су индикатори 0-1 случајне променљиве имамо:

$a_{i,j} = Ex[A_{i,j}] = Pr[A_{i,j} = 1] = Pr[x_i$ је предак $x_j]$, и
$c_{i;l,m} = Ex[C_{i;l,m}] = Pr[C_{i;l,m} = 1] =$
$= Pr[x_i$ је заједнички прекад од x_l и $x_m]$

Утврђивање вероватноћа, а самим тим и очекивања дато је лемом о претку, која у потпуности карактерише везу претка у трипу:

Лема 1 *Нека је T трип за X и нека је $1 \leq i, j \leq n$. Тада под претпоставком да су сви приоритети различити, (x_i) је предак (x_j) у T ако међу свим (x_h) са $h \in [i, j]$, (x_h) има највећи приоритет.*

Доказ. Нека је (x_m) чвор са највећим приоритетом у T и нека је $X' = (x_v | 1 \leq v < m)$ и $X" = (x_\mu | m < \mu \leq n)$. По дефиницији трипа (x_m) ће бити корен и лево и десно подстабло ће бити трипови за X' и $X"$. Јасно је да је карактеризација претка тачна за све парове чворова укључујући корени чвор. Такође је тачна за све парове $x_v \in X'$, као и за $x_\mu \in X"$, они леже у различитим подстаблима и стога нису у вези предак-следбеник. Заиста, највећи приоритет између v и μ реализован је у x_m, а не у x_v или x_μ. Коначно, по индукцији карактеризација је тачна за све парове у X' и $X"$. ■

Као непосредна последица ове леме, произилази следећа лема:

Лема 2 *Нека је T трип за X и нека је $1 \leq l, m, i \leq n$ и $l < m$. Нека је p_v приоритет од x_v.*
Под претпоставком да су сви приоритети различити, (x_i) је заједнички предак за x_l и x_m у T ако:

$$p_i = max\{p_v | i \leq v \leq m\} \quad if\ 1 \leq i \leq l$$
$$p_i = max\{p_v | l \leq v \leq m\} \quad if\ l \leq i \leq m$$
$$p_i = max\{p_v | l \leq v \leq i\} \quad if\ m \leq i \leq n$$

ови услови могу се преписати краће на следећи начин:

$$p_i = max\{p_v | min\{i, l, m\} \leq v \leq max\{i, l, m\}\}$$

Ове леме омогућују анализу величина рандомизираног стабла претраге. Постоје два случаја: уравнотежено и неуравнотежено рандомизирано стабло претраге.

4.1 Неуравнотежен случај

Последње две леме олакшавају извођење вероватноћа $a_{i,j}$ и $c_{i;l,m}$ за претка.

Последица 2 *У неуравнотеженом рандомизираном стаблз претраге x_i је предак x_j са вероватноћом $1/(|i - j| + 1)$ другим речима: $a_{i,j} = 1/(|i - j| + 1)$.*

Доказ. Према леми о претку потребно је да приоритет x_i буде највећи међу приоритетима $[i-j]+1$ између чворова x_i и x_j. У случају неуравнотеженог рандомизираног стабла приоритети су независне непрекидне случајне променљиве са вероватноћом $1/(|i-j|+1)$. ■

Последица 3 *Нека је* $1 \leq l < m \leq n$.
У случају неуравнотеженог рандомизираног стабла очекивања за заједничког претка дата су следећим изразима:

$$c_{i;l,m} = 1/(m-i+1) \quad if\ 1 \leq i \leq l$$
$$c_{i;l,m} = 1/(m-l+1) \quad if\ l \leq i \leq m$$
$$c_{i;l,m} = 1/(i-l+1) \quad if\ m \leq i \leq n$$

што се може записати као:

$$c_{i;l,m} = 1/(max\{i,l,m\} - min\{i,l,m\} + 1)$$

Доказ. Аналогно претходном доказу. ■

Спајањем свега добијају се тачни изрази и приближне форме горњих граница очекивања величина рандомизираног стабла претраге. Изрази укључују хармонијске бројеве дефинисане са $H_n = \Sigma_{1 \leq i \leq n} 1/i$. Лако се изводи да су границе за H_n дате неједнакостима $\ln n < H_n < 1 + \ln n$ за $n > 1$.

Теорема 5 *Нека је* $1 \leq l, m \leq n$ *и нека је* $l < m$. *У неуравнотеженом рандомизираном стаблу претраге са* n *чворова важи:*
(I) $Ex[D(x_l)] = H_l + H_{n+1-l} - 1 < 1 + 2 * \ln n$
(II) $Ex[S(x_l)] = H_l + H_{n+1-l} - 1 < 1 + 2 * \ln n$
(III) $Ex[P(x_l, x_m)] = 4H_{m-l+1} - (H_m - H_l) - (H_{n+1-l} - H_{n+1-m}) - 3 < 1 + 4 * \ln(m-l+1)$
(IV) $Ex[SL(x_l)] = 1 - 1/l$; $Ex[SR(x_l)] = 1 - 1/(n+1-l)$

Доказ. Довољно је користити последицу 1 и укључити вредности из последица 2 и 3. ■

Занимљиво је да је у неуравнотеженом рандомизираном стаблу претраге очекивани број предака чвора x једнак његовом очекиваном броју потомака. Главна разлика између случајних променљивих $D(x_l)$ и $S(x_l)$ је да обе имају иста очекивања, али веома различиту расподелу. $D(x_l)$ је строго концентрисана око њених очекивања, а $S(x_l)$ није.

4.2 Уравнотежен случај

Подсетимо се да је у овом случају сваком чвору x_i придружен позитиван број - тежина w_i, и да овакво рандомизирано стабло претраге користи за приоритет чвора x_i максимум w_i независних непрекидних случајних променљивих са истом расподелом вероватноће.

За $i \leq j$ нека w_{ij} означава $\Sigma_{i \leq h \leq j} w_h$ и за $i > j$ дефинише се $w_{i:j} = w_{j:i}$. Нека $W = w_{1:n}$ означава укупну тежину.

Последица 4 *У уравнотеженом рандомизираном стаблу претраге x_i је предак x_j са вероватноћом $w_i/w_{i:j}$, тј.:*

$$a_{i,j} = w_i/w_{i:j}$$

Доказ. Према леми о претку потребно је да приоритет чвора x_i буде највећи међу приоритетима између чворова x_i и x_j. Ово значи да једна од w_i променљивих за x_i треба да буде највећа међу w_{ij} променљивама за чворове између x_i и x_j. Пошто су ове променљиве идентично дистрибуиране то ће се десити са назначеном вероватноћом. ■

Последица 5 *Нека је $1 \leq l < m \leq n$. У случају уравнотеженог рандомизираног стабла претраге очекивања за заједничког претка дата су следећим формулама:*

$$\begin{aligned} c_{i;l,m} &= w_i/w_{i:m} \quad if\ 1 \leq i \leq l \\ c_{i;l,m} &= w_i/w_{l:m} \quad if\ l \leq i \leq m \\ c_{i;l,m} &= w_i/w_{l:i} \quad if\ m \leq i \leq n \end{aligned}$$

Доказ. Доказ је аналоган претходном само базиран на леми о заједничком претку. ■

Теорема 6 *Нека је $1 \leq l, m \leq n$ и нека је $l < m$. У уравнотеженом рандомизираном стаблу претраге са n чворова и укупном тежином W важе следећа очекивања:*

$$\begin{aligned} &\textit{(I)}\ Ex[D(x_l)] = \Sigma_{1 \leq i \leq n} w_i/w_{i:l} < 1 + 2 * \ln(W/w_l) \\ &\textit{(II)}\ Ex[S(x_l)] = \Sigma_{1 \leq i \leq n} w_i/w_{i:l} \\ &\textit{(III)}\ Ex[P(x_l, x_m)] = 1 + \Sigma_{1 \leq i < l}(w_i/w_{i:l} - w_i/w_{i:m}) \\ &\qquad\qquad + \Sigma_{l \leq i < m}(w_i/w_{l:i} + w_i/w_{i:m} - 2w_i/w_{l:m}) \\ &\qquad\qquad + \Sigma_{m < i \leq n}(w_i/w_{m:i} - w_i/w_{l:i}) \\ &\qquad\qquad < 1 + 2 * \ln(w_{l:m}/w_l) + 2 * \ln(w_{l:m}/w_m) \\ &\textit{(IV)}\ Ex[SL(x_l)] = \Sigma_{1 \leq i < l}(w_i/w_{i:l-1} - w_i/w_{i:l}) < 1 + \ln(1 + (w_l/w_{l-1})) \\ &\qquad Ex[SR(x_l)] = \Sigma_{l < i \leq n}(w_i/w_{l+1:i} - w_i/w_{l:i}) < 1 + \ln(1 + (w_l/w_{l+1})) \end{aligned}$$

Доказ. Тачни изрази следе из последица 4 и 5.
Наведене горње границе су последице следеће две неједнакости које се могу лако проверити израчунавањем површине испод криве $f(x) = 1/x$:

$$\alpha/A \leq \ln A - \ln(A - \alpha) \text{ за } 0 \leq \alpha < A \quad (1)$$

$$\alpha/A - \alpha/B \leq (\ln A - \ln(A - \alpha)) - (\ln B - \ln(B - \alpha)) \ , \ 0 \leq \alpha < A \leq B \quad (2)$$

За доказ прве ставке из теореме користимо неједнакост (1) да изведемо:

$$\Sigma_{1 \leq i < l} w_i/w_{i:l} < \Sigma_{1 \leq i < l}(\ln w_{i:l} - \ln w_{i+1:l}) =$$

$$= \ln w_{1:l} - \ln w_l = \ln(w_{1:l}/w_l) < \ln(W/w_l)$$

и аналогно за $\Sigma_{l < i \leq n} w_i/w_l < \ln(w_{l:n}/w_l) \leq \ln(W/w_l)$, додајући у (1) произилази за $i = l$ наведене граница.
Слично, за ставку (IV) користи се израз 2:

$$\begin{aligned}
&\Sigma_{1 \leq i < l}(w_i/w_{i:l-1} - w_i/w_{i:l}) < \\
&< 1 + \Sigma_{1 \leq i < l-1}((\ln w_{i:l-1} - w_{i+1:l-1}) - (\ln w_{i:l} - w_{i+1:l})) = \\
&= 1 + (\ln w_{1:l-1} - w_{l-1:l-1}) - (\ln w_{1:l} - w_{l-1:l}) < \\
&< 1 + \ln w_{l-1:l} - \ln w_{l-1:l-1} = \\
&= 1 + \ln(1 + w_l/w_{l-1})
\end{aligned}$$

За доказ ставке (III) користи се неједнакост (2), и ограничава се прва сума са $\ln(w_{l:m}/w_l)$, и трећа са $\ln(w_{l:m}/w_m)$. Средња сума може бити подељена на три дела. Трећи део једнак је -2. Коришћењем неједнакости (1), први део је ограничен са $1 + \ln(w_{l:m}/w_l)$, а други део са $1 + \ln(w_{l:m}/w_m)$. Заједно ово даје наведену границу. ■

Поглавље 5

Анализа операција рандомизираног стабла

У овом одељку биће разматране разне операције неуранотеженог и уравнотеженог рандомизираног стабла и изведена очекивања ефикасности доказујући границе из теорема 2 и 3.

5.1 Претрага

Успешно тражење за чвор x у трипу T почиње у корену T и користећи кључ чвора x прати се путања од корена T до чвора x. Јасно је да је потребно време таквог тражења пропорционално дужини путање или другим речима броју предака чвора x. Према томе очекивано време тражења по x пропорционално је очекиваном броју његових предака, који је $O(\log n)$ у неуравнотеженом случају и $O(1 + \log(W/w(x)))$ по теоремама 5 и 6. Неуспешно тражење за кључ завршава се између кључева узастопних чворова x^- и x^+ за очекивано време $O(\log n)$ у неуравнотеженом стаблу, односно $O(\log(W/min\{(w(x^-), w(x^+))\}))$ за уравнотежен случај, после којег се такво тражење мора прекинути било у x^- или у x^+(случајеви када не постоји једно од x^- или x^+ се игноришу).

5.2 Уметање и брисање

Чвор се умеће у трип тако што се прво убаци у одговарајућу позицију у листу која је добијена (неуспешним) тражењем по кључу чвора. Онда се чвор пење у стаблу све док отац нема мањи приоритет од њега. Јасно је да број ових ротација може бити највише као дужина путање управо пређене претраге. Такво време уметања пропорционално је времену

потребном за неуспешно тражење које је у просеку једнако $O(\log n)$ очекивано у случају неуравнотеженог стабла, тј. $O(\log(W/min\{w(x^-), w(x^+)\}))$ у уравнотеженом стаблу, где су x^- и x^+ претходник и следбеник чвора x у резултујућем стаблу.
Операција брисања захтева исто време као уметање, јер је у суштини то инверзан процес. Прво се лоцира жељени чвор у стаблу уз помоћ кључа. Затим се спушта у стаблу док не буде у листу, након чега се одсеца. При ротацији наниже, одлука о левом или десном ротирању зависи од приоритета актуелне деце, оно са већим приоритетом се мора попети горе.
Оно што је још занимљиво је да се ротације јављају и током ажурирања. Како је брисање инверзан поступак уметању, довољно је анализирати број ротација при брисању.
Нека је x чвор који треба обрисати из трипа T. Иако не може да се каже које ће ротације бити примењене приликом спуштања(леве или десне), може се рећи колико ће их бити. Њихов број је сума дужине десне кичме левог подстабла x и леве кичме десног подстабла x. Последица чињенице је да лева ротација за x има ефекат смањивања леве кичме десног подстабла за један, и десна ротација има ефекат смањивања десне кичме левог подстабла за један. Према томе знамо из теорема 5 и 6 да је очекивани број ротација по ажурирању мањи од 2 у неуравнотеженом стаблу, а мањи од $2 + \ln(1 + w(x)/w(x^-)) + \ln(1 + w(x)/w(x^+)) = O(1 + \log(1 + w(x)/w(x^-) + w(x)/w(x^+)))$ за уравнотежено стабло.

5.3 Уметање и брисање са ручкама

Уколико постоји ручка(показивач) на чвор који се брише из трипа, уклања се потреба за тражењем истог. Једино су потребне ротације за спуштање у лист и одсецање листа, за које је очекивано време извршавања $O(1)$ у неуравнотеженом стаблу, односно $O(1+\log(1+w(x)/w(x^-)+ w(x)/w(x^+)))$ у уравнотеженом.
Слично ако знамо позицију листа где се умеће нови чвор x, онда после уметања новог листа потребно је извршити ротације за померање чвора на горе, чије очекивано време извршавања је управо споменуто. Ако је доступан само показивач на x^- и x се не може уметнути као лист у x^- док не постоји десно подстабло, онда настаје додатни трошак за лоцирање x^+. У овом случају x^+ се налази на дну леве стране десног подстабла x^- и његово проналажење кошта колико и прелажење леве стране. Ова операција очекивано кошта $O(1)$ у неуравнотеженом стаблу, и $O(1 + \log(1+w(x^-)/w(x^+)))$ у уравнотеженом, што као резултат даје очекивано

време $O(1)$ и $O(1 + \log(1 + w(x)/w(x^-) + w(x)/w(x^+) + w(x^-)/w(x^+)))$ у одговарајућим случајевима.
За разлику од обичног уметања, уметање са ручкама захтева да показивач на оца буде доступан, како би могле да се изврше ротације навише. Слично и за брисање са ручкама, показивач на оца је неопходан. Од тренутка када се чвор који се брише спушта наниже, мора да зна свог оца y, како би се одређивање показивача сина y могло ресетовати.

5.4 Претрага са показивачем

У овој претрази претпоставка је да постоји ручка или показивач на чвор x у стаблу, и користећи ову информацију желимо да пронађемо чвор y са датим кључем k. Без општег губитка претпостављамо да је $x = x_l$ и $y = x_m$ тако да је $l < m$. Најприроднији начин да се изврши таква претрага је да се прати јединствена путања од x_l до x_m у стаблу. Очекивана дужина овакве путање дата је у теоремама 5 и 6 за неуравнотежена и уравнотежена рандомизирана стабла претраге. Како год, ствари нису баш једноставне док не буде јасно да претрага може прећи путању од x_l до x_m у времену пропорционално њеној дужини. Таква претрага би подразумевала генерално, пењање навише у стаблу са почетком из x_l до најнижег заједничког предка u од x_l и x_m и затим спуштање до x_m. Али како знати да је претрага стигла до u?

Ако је претрага током успона ка кореном чвору достигла чвор z чији је кључ већи од кључа по коме се претражује, онда је претрага отишла предалеко. Жељено u је било или последњи чвор на пређеном путу који је постигнут преко његовог левог сина, ако такав чвор постоји, или стартни чвор x. Овај вишак пута између u и z може бити много дужи него пут између x и y, и прелажење овог пута може бити доминантно у времену извршавања претраге са показивачем.

Постоји неколико начина да се реши проблем са вишком пута. Једно је да се тврди да је очекивано да је такав пут константне дужине. Међутим испостави се да је ово тачно само за неуравнотежен случај. Друге методе подразумевају чување додатних информација у стаблу које дозвољавају пречицу за вишак путање или уклањају потребу за прелазак таквих.

Сада ће бити укратко објашњена метода додавања **допунског показивача**. Дефинише се леви родитељ чвора v на путањи од v до кореног чвора, као први чвор чији је кључ мањи од кључа чвора v, и десни

родитељ као први чвор на путањи са кључем већим од кључа чвора v. Другачије речено чвор је десни родитељ свих чворова на десној страни његовог левог подстабла, и леви родитељ је родитељ свих чворова на левој страни његовог десног подстабла. За сваки чвор се поред показивача на два сина, чувају и показивачи на левог и десног родитеља, односно nil ако такав чвор не постоји. Током ротација, или додавања или одсецања листова, цена одржавања ових показивача је константна. Тако да ово нема значај на пораст времена ажурирања.

Претрага са показивачем креће из x тражећи показивач на десног родитеља или док се не наиђе на показивач на nil или на следећи чвор са кључем већим од k. У тренутку када претрага пронађе најмањег заједничког предка u од x и y, и користећи кључ k, претрага проналази пут од u до y пратећи показиваче на синове на уобичајен начин. Треба имати на уму да је број веза које је претрага прешла највише за 1 већа од дужине путање између x и y и да може бити много мањи због пречица које пружају показивачи на десног родитеља.

Са сваким чвором u у стаблу чувају се и **допунски кључеви** $min(u)$ и $max(u)$, најмањи и највећи кључ чвора подстабла чвора u. Са овим информацијама може се знати у константном времену да ли чвор може бити у подстаблу чвора u: његов кључ мора бити у распону између $min(u)$ и $max(u)$. Заједнички предак од x_l и x_m је онда први чвор z на путањи од x_l до чвора са $min(z) \leq k \leq max(z)$.

Цена одржавања информација $min(u)$ и $max(u)$ током ротација је константна. Међутим, када се дода лист приликом уметања или одсече приликом брисања ове информације је потребно ажурирати на почетку пута ка чвору. Због прецизности размотримо случај брисања листа x. Чворови за које се min информација мења су тачно они чворови на левој страни десног подстабла предка чвора x. Информација max треба бити прилагођена за све чворове на десној страни левог подстабла следбеника x. Очекиване дужине ових страна подстабла дате су у теоремама 5 и 6. Оне морају бити додате на очекивано време ажурирања које је асимптотски ирелевантно за случај неуравнотеженог стабла, али може да буде доминантан фактор у случају уравнотеженог стабла.

Коначно да би овај аргумент био примењив на чворове са најмањим и највећим кључевима у стаблу, потребно је одржавати трип са два чвора

која се никад не уклањају, један са кључем $-\infty$ и други са ∞ (оба са случајним приоритетима).

Ослањање на кратки вишак путање

Сада ће бити реч о методу који прелази део путање који је вишак и ослања се на чињеницу да је према очекивањима тај део кратак, бар за неуравнотежено стабло. Предност ове методе је што је потребно чувати само по један показивач родитеља по чвору. Међутим потребно је чувати још један додатни бит по чвору.

Нека је u најмањи заједнички предак између $x = x_l$ и $y = x_m$ и нека је десни родитељ од u чвор z крајњи чвор на делу путање који је вишак. *Вишак чворови* су чворови који се налазе између u и z. Нека је $u = x_v$ за неко v где је $l \leq v \leq m$ и $z = x_j$ за неко j где је $m \leq j \leq n$ и сваки *вишак чвор* мора бити неко x_i где је $1 \leq i < l$. Очекивани број оваквих чворова процењује се на основу уведене променљиве X_i, за свако i, $1 \leq i < l\ 0-1$ која означава да ли је x_i *вишак чвор* или не и сабирањем очекивања. $Ex[X_i]$ је вероватноћа да је x_i *вишак чвор* који се представља као сума вероватноћа раздвојених догађаја E_ij тако да је x_i *вишак чвор*, а z је x_j. За E_ij који се деси, x_j мора бити десни родитељ x_i и u, или другачије речено x_i и u на десној страни левог подстабла x_j. Ово се дешава када на раздаљини између i и j чвор x_j има највећи приоритет, а x_i има следећи највећи приоритет и ако u има највећи приоритет између l и $j-1$. По дефиницији u последњи услов се може преформулисати тако да највећи приоритет у опсегу l и $j-1$ збије на опсег од l до m. По претпоставци да су сви приоритети независни и идентично расподељени следи:

$$Pr[E_{ij}] = \frac{1}{j-i+1} \cdot \frac{1}{j-1} \cdot \frac{m-l+1}{j-l}$$

и очекивани број *вишак чворова* је

$$\Sigma_{1 \leq i < l} Ex[X_i] = \Sigma_{1 \leq i < l} \Sigma_{m < j \leq n} Pr[E_{ij}] = \Sigma_{1 \leq i < l} \Sigma_{m < j \leq n} \frac{1}{j-i+1} \cdot \frac{1}{j-1} \cdot \frac{m-l+1}{j-l}$$

Претходна анализа односи се на постојање z, тј. најмањи заједнички предак u мора имати десног родитеља. Ово не мора да буде случај: u може лежати на десној страни стабла. У овом случају наивна претрага са показивачем ће наставити да иде навише поред u док не дође до чвора. Да се ово спречи потребно је да се за сваки чвор чува додатна информација да ли је на десној или левој страни. Са овом информацијом претрага може да престане да се пење чим се открије десна страна стабла и на тај начин биће пређен пут само између x и y.

5.5 Брза претрага са показивачем

Претпоставка је да постоји показивач на чвор x и један на z и потребно је брзо пронаћи чвор y са кључем k. Нормална ствар је стартовати претрагу и из чвора x и из z и пратити их паралелно док прва не пронађе y. Ако су X и Z дужине од x и z до y, односно према претходном одељку време потребно за такву паралелну претрагу било би пропорционално $min\{X, Z\}$ и очекивано време би било пропорционално $Ex[min\{X, Z\}] \leq min\{Ex[x], Ex[z]\}$.

Ово се може искористити на следећи начин: увек одржавати показиваче на $Tmin$ и $Tmax$, најмањи и највећи кључеви чворова у стаблу. Ако претрага треба да крене из чвора $x = x_l$ за неки чвор $y = x_m$ и кључ чвора y је већи од кључа x, треба паралелно покренути брзу претрагу и из чвора $Tmax = x_n$. Очекивано време ове претраге је пропорционално минимуму очекиване дужине путање од x и $Tmax$ до y, што је по теореми 5 $O(min\{\log(m - l + 1), \log(n - m + 1))\}$.

5.6 Подела и спајање

Два трипа T_1 и T_2 са кључевима у T_1 мањим од кључева у T_2 могу бити спојена креирањем новог стабла T са кореном r чији је леви син T_1 а десни T_2 и затим брисањем r из T.

Стабло T се креира за константно време. Брисање чвора r, као што смо раније разматрали захтева време пропорционално збиру дужине десне стране стабла T_1 и дужине леве стране T_2 тј. збиру дубина највећег кључа у T_1и најмањег у T_2. Према теореми 5 ово очекивање је $O(\log m + \log n) = O(\log max\{(m, n)\})$, ако су T_1 и T_2 неуравнотежена рандомизирана стабла претраге са m и n чворова. За уравнотежено рандомизирано стабло ова очекивана граница је $O(1+\log \frac{W_1}{w(T_1 max)} + \log \frac{W_2}{w(T_2 min)})$, при чему је са W_i означена укупна тежина T_i.

Подела трипа T на трип T_1 који има све кључеве мање од k и на трип T_2 са кључевима већим од k је у суштини обрнут процес спајању 2 трипа: чвор x са кључем k и бесконачним приоритетом се уметне у стабло; због бесконачног приоритета он постаје чвор; његово лево и десно подстабло ће бити T_1 и T_2. За ово уметање прво се покреће претрага из корена како би се одредила позиција листа за кључ k у ком ће се додати чвор x и онда се низом ротација он пење на позицију корена. Овде је дужина путање

претраге једнака броју изведених ротација што је уствари збир дужина десне стране T_1 и дужине леве стране T_2. Време потребно за извршење ове операције пропорционално је збиру ових дужина, стога је очекивано време ове операције исто као и време спајања T_1 и T_2.

5.7 Брза подела

Да би се убрзала описана процедура, потребно је смањити време претраге за одговарајућу позицију као и смањити број ротација које следе након претраге. Претпоставка је да је величина коначног T_1 мала. Уз ову претпоставку има смисла урадити претрагу са показивачем за проналажење одговарајуће позиције листа у T, која креће из $Tmin$, чвором са најмањим кључем у T. Нека је z најмањи заједнички предак $Tmin$ и листа додатог за чвор x. Евидентно је да се z налази на путањи између $Tmin$ и x и да z мора бити на левој страни у T. При пењању чвора x у стаблу, операција брзе поделе се може зауставити чим z постане леви син чвора x. Тренутно лево подстабло чвора x ће садржати тачно све чворове из T са кључевима мањим од кључа k по коме се деле два трипа и формира се жељени T_1. Стабло T_2 добија се једноставном заменом подстабла са кореном x са његовим десим подстаблом.

Време потребно за све ово је пропорционално дужини путање L у оригиналном стаблу T између $Tmin$ и листа додатог за x. У неуравнотеженом стаблу очекивана вредност L је по теореми 5 $O(\log m)$ где је m број чворова који су се попели у T_1. Ово и није баш добро у случају да T_1 представља већи део од T. У том случају би било боље симетрично покренути брзу претрагу из $Tmax$, чвором у T са највећим приоритетом и одредити путању од $Tmax$ до x. Очекивана дужине R те путање, а самим тим и целе операције поделе је $O(\log n)$ где је n величина стабла T_2. Наравно да се не може знати унапред да ли је мање T_1 или T_2. Овај проблем се решава тако што се паралелно извршавају претраге и прати се њихова цена док се једна од њих не заврши успешно. Очекивано време за ово је $O(min\{(L,R)\})$. Према $Ex[min\{(L,R)\}] \leq min\{(Ex[L], Ex[R])\}$ укупно очекивано време операције брзе поделе је $O(min\{(\log m, \log n)\}) = O(\log min\{(m,n)\})$.

За уравнотежен случај довољно је на основу теореме 6 посматрати да је очекивано L $O(1+\log(\frac{W_1}{w(T_1 min)}+\frac{W_1}{w(T_1 max)}))$ где је $T_1 max$ чвор са највећим кључем у T_1, што је заправо x^- тј, предак x-а у T. Међутим, ово није сасвим тачно све док нови лист x није нужно син чвора x^-, али може

бити и син чвора x^+ који је наследник чвора x. У том случају путања у T од x^- до x^+ која треба бити додатно пређена након проналажења x^- може бити прилично дуга, заправо $O(1+\log(1+\frac{w(x^-)}{w(x^+)})$. Овај трошак са додатном путањом се може избећи ако се користи више простора за чување показивача на претходника и следбеника за сваки чвор у стаблу.

5.8 Брзо спајање

У брзом спајању рандомизирано стабло покушава да инвертује операцију брзе поделе. Почиње у $T_1 max$ и $T_2 min$, и прелази десну страну RS стабла T_1 и леву страну LS стабла T_2 при спајању, све док не пронађе чвор x у RS чији је приоритет већи од приоритета корена T_2, симетрично важи и за другу страну (док се не пронађе чвор y у LS чији приоритет је већи од корена T_1). У првом случају подстабло T_x стабла T_1 мења се спојеним стаблима T_x и T_2 добијеног помоћу нормалног алгоритма спајања, а корене резултујућег стабла је корен стабла T_1 (симетрично важи и за други случај где корен T_2 постаје корен резултата). Ова операција је у ствари инверзна операцији брзе поделе, с тим да је једина разлика да овде није потребно користити претрагу са показивачима. Зато је време потребно за спајање T_1 и T_2 горње ограничено временом брзе поделе стабла T на T_1 и T_2.

5.9 Одсецања

Нека су x и y два чвора трипа T са n чворова, и претпоставка је да постоји показивач на бар један од њих. Треба из трипа T издвојити трип T' који садржи све чворове d чији су кључеви између (укључујући и) кључева чворова x и y, остављајући све остале чворове у T. Прво треба показази да је ово изводљиво у времену пропорционалном $P(x,y)$, дужини путање од x до y и $SL(x)$ и $SR(y)$, дужине путања десне стране левог подстабла x, и леве стране десног подстабла y. По теореми 5 добија се $O(\log d)$.

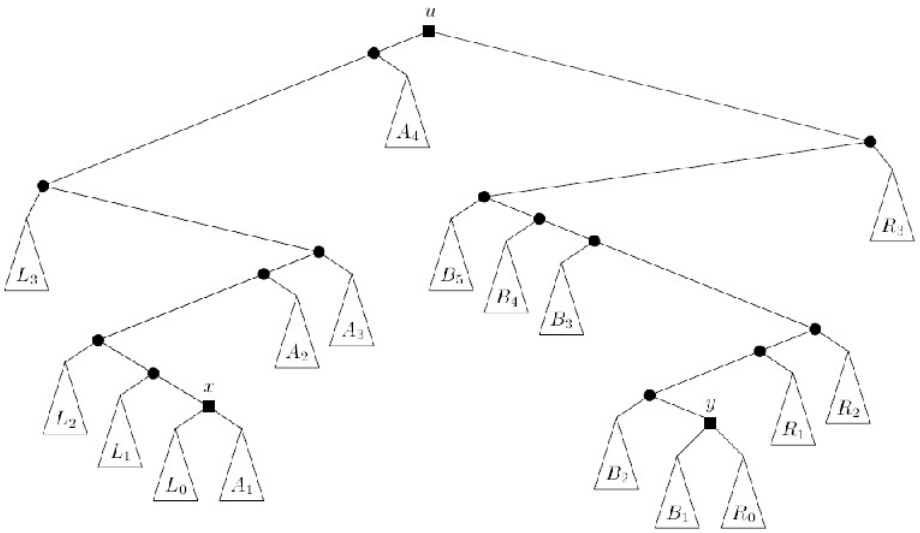
Помоћу претраге са показивачем прво се одреди путања која повезује x и y и са тим и њихов најмањи заједнички предак u у времену $O(P(x,y))$. Нека је T_u подстабло са кореном u. Довољно је показази како одсећи T' из T_u. Нека је LL низ чворова на путањи од x до u са кључевима мањим од кључа x и нека је A низ преосталих чворова на путањи не укључујући u. Симетрично се дефинише RR, B за путању од y до u. Сада жељени

трип T' има корен u, лево подстабло се добија низањем чворова из A заједно са десним подстаблима; десно подстабло се добија као лево само са чворовима из B. Јасно је да се T' може формирати за $O(P(x,y))$.

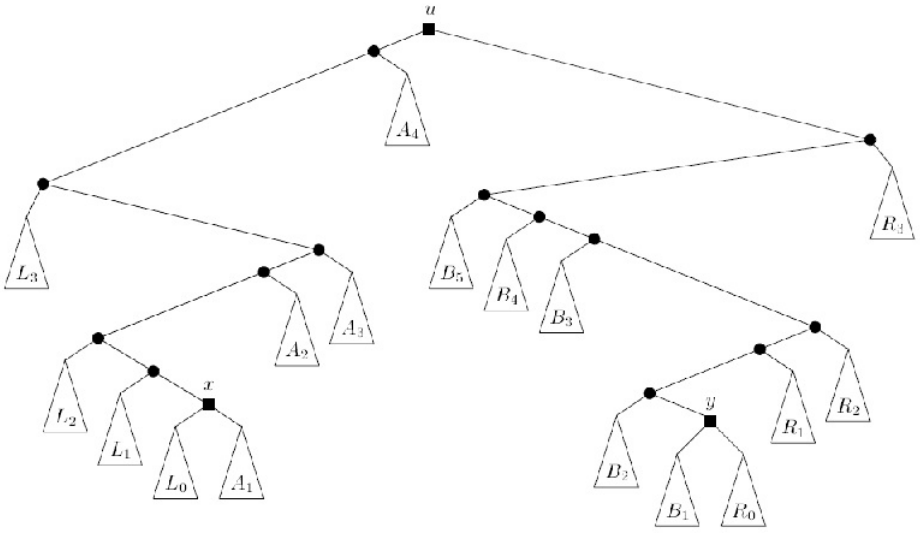
Остали делови од T_u могу се вратити у трип на следећи начин: формира се трип L додавањем заједно чворова у LL са њиховим левим подстаблима, додавањем левог подстабла x на дно десне стране. Симетрично се формира и трип R. Остатак T_u се конструише додавањем чланова LL у стабло L и RR у R. Јасно је да се L и R могу конструисати за $O(P(x,y))$. У претходном делу текста показано је да је време операције спајања пропорционално суми дужина десне стране L и леве стране R. Али ова сума је највише $P(x,y) + SL(x) + SR(y)$.

Исецање T' из T такође може бити изведено у очекиваном времену $O(\log(n-d))$ користећи следеће методе: подела трипа T на трипове L и T", где L садржи све чворове чији су кључеви већи од кључа чвора x; подели се трип T" на жељени T' и R, где R садржи све чворове са кључевима већим од кључева чвора y; на крају се спајају L и R како би формирали остатак трипа. Коришћењем метода брзе поделе и спајања(обичан алгоритам) ово се може извршити у очекиваном времену од $O(\log(n-d))$.

Наравно, d се не може знати унапред. Због тога се поставња питање коју методу користити. Ово се може решити на следећи начин: прати се паралелно извршавање брзе претраге од x до y, и од $Tmin$ до x праћено брзом претрагом од $Tmax$ до y. Метод који се први изврши ће бити коришћен.



Слика 5.1 *Подстабло T_u пре исецања*



Слика 5.2 *Стабла L, T' и R*

Поглавље 6

Програмска реализација

У овом поглављу описана је програмска реализација бинарне структуре трип у програмском језику *C++*. У другом делу овог поглавља приказани су резултати експеримената.

6.1 Програмска реализација трипа

Програмска реализација трипа и експерименти имплементирани су у програмском језику *C++* и у *DevC++* окружењу. Имплементирана је структура која представља један чвор и функције: десна и лева ротација, функција за креирање новог чвора, уметање чвора, тражење чвора са задатим кључем, брисање чвора, испис структуре, функција за генерисање случајног броја, функија за креирање трипа од датог текстуалног фајла.

Структура која представља један чвор садржи кључ, приоритет и показиваче на лево и десно подстабло. Функција за леву ротацију (функција за десну ротацију аналогна је левој ротацији) прима као улазни параметар показивач на корен стабла односно подстабла и дата је следећим кодом:

```
template <typename T>
treap<T>* LeftRotation(treap<T>* aK2){
    treap<T>* lk1 = aK2->rChild;
    aK2->rChild = lk1->lChild;
    lk1->lChild = aK2;
    return lk1;
}
```

Функција за креирање новог чвора прима вредност за кључ као улазни параметар, док се као приоритет узима као случајни број. Функција за уметање чвора као улазне параметре прима вредност кључа $aKey$ на основу кога креира нови чвор и уноси у структуру чији је корен улазни параметар $aRoot$. Функција за проналажење чвора као улазне параметре прима корен структуре у којој се тражи чвор и вредност кључа који се претражује. Код брисања чвора као улазни параметри прослеђују се корен структуре из које се брише чвор и вредност кључа чвора који се брише.

Функција $timeMesurments$ која за унети број чворова(који је неки степен двојке) рачуна насумичне вредности кључева трипа и уноси их у низ. Након уноса вредности кључа у низ, уноси исте вредности у трип и тако бележи време почетка операције уметања. Након завршетка операције уметања елемената рачуна се укупно време потребно за уметање елемената. Из тако формираног стабла почиње брисање елемената и то свих елемената који су претходно били унети. Низ $arrayOfKeys$ се користи као низ свих чворова који су претходно унети у трип па самим тим и свих елемената који ће бити обрисани. Бележи се почетак операције брисања као и крај како би се израчунало укупно време које је потребно да се ова операција изврши. Функција је дата псеудо кодом:

```
void timeMesurments(){
    insertNumberOfNodes();
    allocateMemoryForKeys();
    generateRandomValues();
    getCurrentTime();
    insertValuesInTreap();
    calculateTime();
    getCurrentTime();
    deleteFromTreap();
    calculateTime();
}
```

Функција $insertFromFile$ креира трип за дати текстуални фајл. Из текстуалног фајла се извлаче и скупљају речи и броје се њихова појављивања. Речи из фајла су представљене као кључеви у трипу, док учесталост појављивања једне речи представља њен приоритет у структури. Најпре се од фајла креира мапа која се састоји од парова речи и броја њеног појављи-вања, а затим се од мапе креира трип. Након тога се све

речи из трипа претражују и региструје се време утрошено на операцију претраге. Функција је дата следећим псеудо кодом:

```
void insertFromFile(){
    openFile();
    createMapFromFile();
    createTreapFromMap();
    getCurrentTime();
    searchValuesInTreap();
    calculateTime();
    closeFile();
}
```

6.2 Резултати експеримената

У наставку текста наведени су резултати експеримената са трипом. Резултати су представљени табеларно и графички. Први експеримент се састоји из серије мерења времена за операције уметања и брисања у трип и када се користи стандардан скуп из СТЛ. Прво је урађена серија мерења времена за уметање чворова у трип. Број чворова који се умеће је степен двојке, па су бележена времена за уметање n чворова где је n $2^k, k = 15, 16, ...19$. Након тога исто је поновљено и за брисање. Добијена времена у милисекундама дата су у табели 6.1:

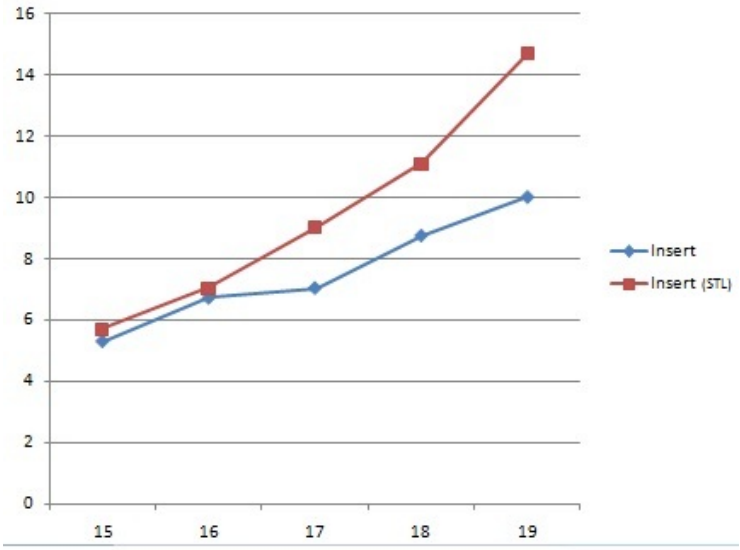
Број чворова	Уметање	Уметање(СТЛ)	Брисање	Брисање(СТЛ)
2^{15}	40	53	19	32
2^{16}	107	132	47	95
2^{17}	134	523	203	313
2^{18}	439	2191	288	1207
2^{19}	1051	26342	985	75610

Табела 6.1: Времена извршавања операција уметања и брисања у трип и стандардног скупа.

На сликама 6.1 и 6.2 дат је графички приказ резултата експеримената. Ради боље прегледности узете су логаритамске вредности за добијене резултате мерења.

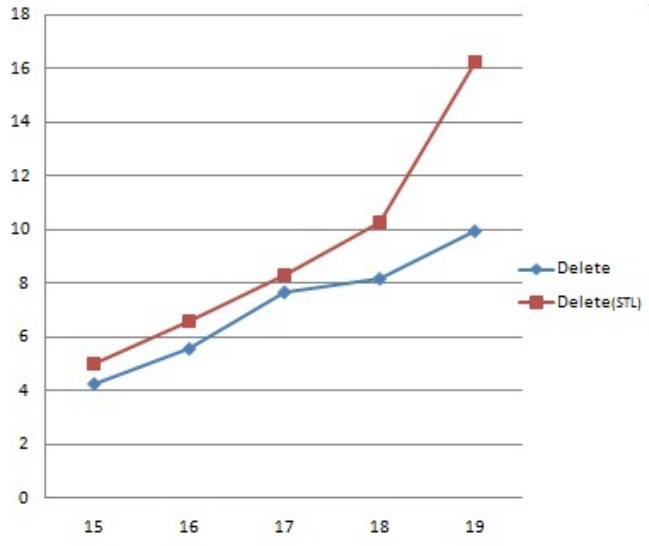
На слици 6.1 приказане су операсације уметања у трип и уметања када се користи стандардни скуп из СТЛ. На графику се види да је у свакој тачки мерења време потребно за операцију уметања у трип мање него када се користи стандардни скуп из СТЛ.

На слици 6.2 приказане су операсације брисања из трипа и брисања када се користи стандардни скуп из СТЛ. Као и код операције уметања и код операције брисања у свакој тачки мерења време потребно за операцију брисања из трипа је мање него када се користи стандардни скуп из СТЛ.



Слика 6.1 *Графички приказ резултата мерења операције уметања у трип и када се користи стандардни скуп СТЛ*

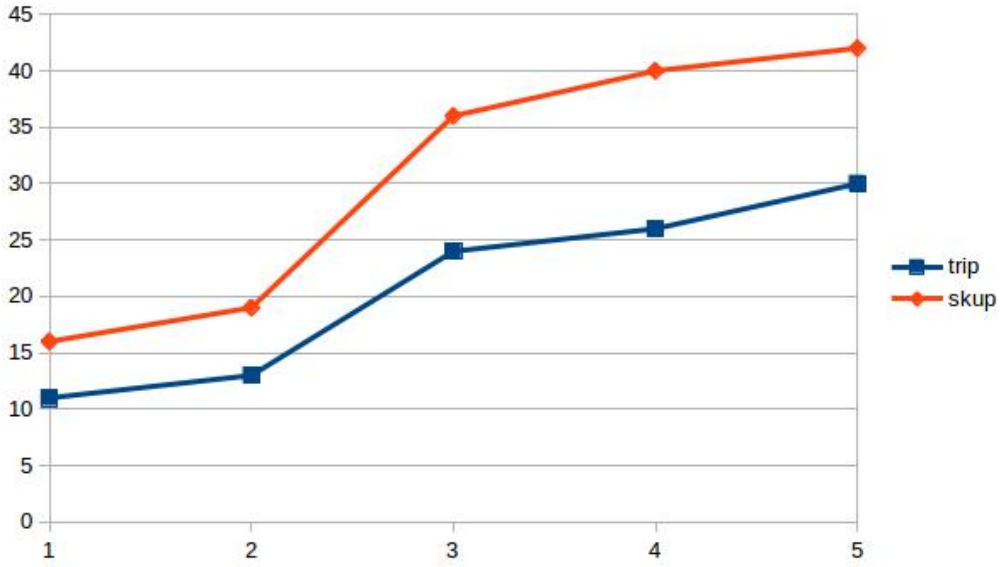
У другом експерименту полази се од великог текстуалног фајла. Прво треба формирати речник од текстуалног фајла преко трипа и скупа. Проналазе се све речи у фајлу и броје се њихова појављивања. У случају трипа број појављивања речи представља приоритет, а сама реч је кључ чвора. Скуп је имплементиран преко *multiset*-а. У самом експерименту мери се време потребно за проналажење свих речи у трипу, и време потребно за проналажење свих речи у скупу. У табели 6.3 представљени су добијени резултати изражени у милисекундама. Мерења су извршена у 5 серија. У свакој серији коришћена је различита дужина фајла, где су дужине фајла $2^k, k = 15, 16, ...19$. На слици 6.3 графички је приказан добијени резултат експери-мента.



Слика 6.2 *Графички приказ резултата мерења операције брисања из трипа и када се користи стандардни скуп СТЛ*

Дужина фајла	Трип	Скуп
2^{15}	11	16
2^{16}	13	19
2^{17}	24	33
2^{18}	27	40
2^{19}	20	42

Табела 6.3: Времена извршавања операције претраге речи у речнику помоћу трипа и скупа.



Слика 6.3 *Графички приказ резултата мерења операције претраге речи у речнику*

Поглавље 7

Закључак

У овом раду описана је бинарна структура података трип и операције за рад са трипом. Трип је ефикасна структура која чува кључеве и приоритете. Подржане операције су: тражење елемента са задатим кључем, уметање новог, брисање задатог, подела на два мања и спајање у један трип. Главна примена трипа је рандомизирано стабло претраге. Како би се убрзале операције рандомизираног стабла претраге користе се покази-вачи или ручке на чвор.

Имплементирана је структура трипа са операцијама објашњеним у раду и урађен експеримент мерења времена потребног за уметање и брисање чворова у трип. На време трајања операција утиче или позиција на којој се налази чвор који се брише или позиција на којој чвор треба да буде после уметања.

Трип се може користити у раду са динамичким низовима. Овакав трип који се назива имплицитни трип подржава операције: убацивање елемента на било коју позицију, брисање елемента са било које позиције, поделу низа на два мања, спајање два низа у један већи. Идеја оваквог трипа је да се као кључ чува индекс елемента, а не његова вредност.

Литература

[1] Raimund Seidel, Cecilia R. Aragoon *Randomized Search Trees*. Computer Science Division University of California Berkeley - 30th Annual Symposium on Foundations of Computer Science, 1989.

[2] Asgeir Davidsson, David Roggers *Treaps*.

[3] http://www.cs.cornell.edu/courses/cs312/2003sp/lectures/lec26.html

[4] http://stackoverflow.com/questions/2509679/how-to-generate-a-random-number-from-within-a-range

[5] http://stackoverflow.com/questions/801740/c-how-to-draw-a-binary-tree-to-the-console

[6] https://stackoverflow.com/questions/26109971/counting-same-string-word-in-a-text-file-in-c