

Próf 2019

Tölurnar 1 til 9 eru settar á stafla og teknar af með blöndu af push og pop. Hver af eftirfarandi röðum getur aldrei komið fyrir þegar talan er prentuð út í hvert skipti sem kallað er á pop: 214359678

Hver af eftirfarandi netareikniritum getur ekki svarað því hvort óstefnt net (Undirected Graph) sé samanhangandi eða ekki (Connected): Topological sort

Hvert af eftirfarandi röðunaraðferðum hefur sömu tímaflækju fyrir besta og versta tilfelli og er stöðug (stable): Merge sort

Reiknirit hefur tímaflækjuna $T(n) < n^2$. Hver af eftirfarandi staðhæfingum er ekki rétt túlkun á þessari staðreynd: Tímaflækjan er n^2 fyrir öll n.

Hver er tímaflækja á eftirfarandi aðferð, sem fall af lengd strengsins N:

```
public static String rev(String s)
String r = ; || int N = s.length();
for (int i = 0; i < N; i++)
    r = s[i] + r; || return r;
```

Eftir i umferðir er r með i+1 staf. Aðgerð númer i kostar því $O(i)$ aðgerðir því það þarf að leggja saman tvo strengi, af lengd i og 1. Heildarfjöldi aðgerða verður því $1+2+3+\dots+N = N^2$ ($O(N^2)$)

Gefið er eftirfarandi fall sem finnur M stærstu gildi í fylki af stærð N:

```
public static Iterable<Integer>
topM(int[] a, int M)
int N = a.length;
MinPQ<Integer> pq = new
MinPQ<Integer>(M+1);
for (int i = 0; i < N; i++)
    if (pq.size() < M) pq.insert(a[i]);
    if (pq.min() <= a[i])
        pq.insert(a[i]); pq.delMin(); || return pq;
```

Hver er versta og besta tímaflækja fyrir reikniritið. Rökskýðið svarið og gefið dæmi um eitt inntak sem nær bestu tímaflækju og annað sem nær verstu tímaflækju. Athugið að svarið getur verið fall af bæði M og N:

Í besta tilfelli er aldrei (eða aðeins einu sinni) farið inn í seinni íf setninguna, þ.e. þegar a er vaxandi. Þá er bara farið M sinnum í fyrri íf setninguna. Hver aðgerð þar kostar $\log(i) \leq \log(M)$ aðgerðir, samtals $M \log(M) + N$. Reyndar er hægt að

vera með fylkið M, M-1, ..., 1, M+1, M+2, ..., N og þá kostar innsetningin bara $O(1)$ og tímaflækjan verður $M+N$. Í versta tilfelli (t.d. a minnkandi) er alltaf keyrt insert sem kostar þá $\log(M)$ fyrir hverja innsetningu, heildarkostnaður verður þá $N \cdot \log(M)$

Teiknið eftirfarandi gagnagrindur þegar búið er að setja stökin 5,3,8,1,2,6,4,9,7 inn í þessari röð:

(a) Hrógu (Max Heap). Hér þarf bara að teikna lokaniðurstöðu. Teiknið bæði tréð og samsvarandi fylki:

(b) Tvíleitartré (Binary Search Tree). teikna lokaniðurstöðu:

(c) Vinstri hallandi rautt-svart-tvíleitartré (Left Leaning Red Black Binary Search Tree). Þetta er sú útgáfa sem við fórum yfir í námskeiðinu. Teiknið gagnagrinduna eins og hún lítur út eftir hverja innsetningu, táknið rauða leggi með r. Til einföldunar eru fyrstu fjögur trén gefin:

Hakkatafla af stærð 12 með hakkafallinu $h(k) = 3k$ og árekstrar eru leystir með Linear Probing. Hér þarf bara að teikna lokaniðurstöðu:

Næstu tvær spurningar eru óháðar en nota sömu skilgreiningu á Node klasa í tvíleitartré (BST):

```
private class Node || private Key key;
private Value val; || private Node left,
right;
private double weight; || private double
weightSum;
```

Hér er weight jákvæð tala sem táknar þyngd hnúts, weightSum er summan af öllum weight gildum í hluttrénu. Í upphafi eru eingöngu weight gildin gefin en öll weightSum vantar. Klárið

að skrifa en- durkvæma fallið computeWeightSum sem reiknar út rétt gildi fyrir weightSum og geymir gildin í hnútunum.

```
private void computeWeightSum(Node x) || if (x==null) return;
computeWeightSum(x.left);
computeWeightSum(x.right);
weightSum = weight;
if (x.left != null) weightSum += x.left.weightSum;
if (x.right != null) weightSum += x.right.weightSum;
```

Skrifið fall sem tekur inn hnút x og tölu w og skilar minnsta hnút y í hluttrénu undir x sem hefur weightSum a.m.k w (hér er gert ráð fyrir að þegar hafi verið fyllt inn í weightSum með réttum gildum)

```
private Node find(Node x, double w)
if (x==null) return;
if (weightSum < w) return null;
Node y = find(x.left, w);
if (y != null) || return y; || else || return x;
```

Þetta forrit tekur inn net G og ákvarðar hvort netið sé tré (tree). Tré eru net sem hafa enga hringi (cycles) og eru samanhangandi (connected). Athugið að dfs fallið tekur inn auka hnút u sem er hnúturinn sem kallaði á v. Klárið forritið með því að fylla í reitina:

```
public class DepthFirstSearch
private boolean[] marked;
private boolean isTree;
public DepthFirstSearch(Graph G)
marked = new boolean[G.V()];
isTree = true; || dfs(G, 0, 0);
for (int i = 0; i < G.V(); i++)
    if (!marked[i]) isTree = false;
public boolean isTree() || return isTree;
private void dfs(Graph G, int v, int u)
marked[v] = true;
for (int w : G.adj(v)) || if (!marked[w])
    dfs(G, w, v); || else ||
    if (w != u) || isTree = false;
```

Gefið er DAG (Directed Acyclic Graph) G og tveir hnútar s og t í netinu. Lýsið reikniriti sem finnur fjölda ólíkra leiða á milli s og t í netinu G á tíma sem er línulegur í stærð G. Hér þarf ekki að skrifa kóða, en lýsingin þarf að vera nógu nákvæm til að hægt sé að breyta henni í kóða.

Hér þarf að halda utan um fjölda leiða frá t til hvers hnúts v í netinu, geymum

þann fjölda í breytu í hverjum hnút, köllum hana x. Upphafsstillum allar breytur með 0, nema í hnútunum t setjum við fjöldann sem $x=1$. Þar sem netið er DAG byrjum við á að finna grannfræðilega röðun á línulegum tíma. Síðan förum við í gegnum þessa röð og fyrir hvern hnút v þá bætum við v.x við x-gildið hjá öllum hnútum sem v bendir á. Í lokin skoðum við s.x sem er þá fjöldi leiða frá t til s.

Gefið er stefnt vigtað net (Directed Weighted Graph) G, með vigtir w(e) 0 á öllum leggjum og hnúta s og t í netinu. Lýsið reikniriti sem finnur stystu leið með fæsta hnúta á milli s og t. Þ.e. af öllum leiðum milli s og t sem eru jafnlangar þegar vigtirnar eru lagðar saman þá er þessi leið með minnsta fjölda hnúta. Hér þarf ekki að skrifa kóða, en lýsingin þarf að vera nógu nákvæm til að hægt sé að breyta henni í kóða.

Leið 1. Höldum utan um nýja vigt sem er par af double og int. Double hlutinn er w, int hlutinn er fjöldi hnúta á leiðinni. Útfærum samanburð sem ber saman w fyrst og ef það er jafnt þá ber það saman int hlutann. Breytum síðan relax fallinu í Dijkstra til að raða eftir þessari tvennd og uppfæra með því að þegar nýjum legg er bætt við þá er bætt w við double töluna og 1 við leggjatöluna sem samsvarar því að telja minnstu vigt og af þeim sem er minnst, þá minnsta fjölda leggja á leiðinni. Dijkstra með þessari breytingu leysir vandamálið. Leið 2. Finnum stystu leið með venjulegu Dijkstra og finnum svo alla leggi sem eru í stystu leiðar trénu og alla þá sem gætu verið á einhverri stystu leið (þ.e. eru jafnir þegar við keyrum relax). Búum til nýtt net með einungis þessum leggjum, látum allar vigtir á leggjum vera 1 og keyrum Dijkstra aftur.

Spurningar

Hver er keyrslutími á insertion sort þegar fylkið er í öfugri röð? $O(N^2)$

Hver er keyrslutíminn fyrir merge aðgerðina ef listarnir eru af lengd n? $O(N)$

Hver er keyrslutíminn fyrir partition aðgerðina ef listarnir eru af lengd n? $O(N)$

Hver er keyrslutíminn á quick sort þegar inntakið er slembið? $O(N \log(N))$

Hver er keyrslutíminn á select að meðal-

þali? $O(N)$

Hver er keyrslutíminn á insert í forgangsbiðröð ef notuð eru öröðuð fylki? $O(1)$

Hver er keyrslutíminn á insert í forgangsbiðröð ef notuð er kös (e. heap)? $O(\log(N))$

Hvað tekur langan tíma að búa til löglega kös (heap) úr N stökum? $O(N)$

Hver er keyrslutíminn fyrir insert aðgerðina í raðaðan lista af lengd n? $O(n)$

Hvert er verst keyrslutími fyrir leit í BST? $O(n)$

Hver er meðaltals tími fyrir leit í BST? $O(\log(N))$

Hvert er verst keyrslutími fyrir leit í 2-3 tré? $O(\log(N))$

Hvert er verst keyrslutími fyrir leit í hakkatöflu með open addressing og tengdum listum? $O(N)$

Hver er meðaltals keyrslutími fyrir leit í hakkatöflu með open addressing? $O(1)$

Hve lengi tekur að leggja saman N stafi 'a' í streng s í for lykkju? N^2

Hve lengi tekur helmingunarleit að finna stak í N staka lista í versta tilfelli? $O(\log(N))$

Hve lengi tekur helmingunarleit að finna stak í N staka lista í besta tilfelli? $O(1)$

Hve lengi tekur Find aðgerðin með quick union aðferðinni? $O(N)$

Hve lengi tekur Find aðgerðin með weighted-quick union aðferðinni? $O(\log(N))$

Hve lengi tekur Find aðgerðin með weighted-quick union aðferðinni í besta tilfelli? $O(1)$

Hve langan tíma tekur að leysa percolation vandamálið á $N \times N$ borði? $O(N^2)$

Hversu mikið auka minni notar merge sort ef inntakið er af lengd n? $O(N)$

Hvaða röðunaraðferð er ekki stöðug? Selection sort

Hvaða 3 röðunaraðferðir notar introsort? Quicksort, Heapsort og Insertion sort

Hvaða aðferð þurfa lykjar að hafa til að vera notaðir í uppfléttistöflu? compareTo eða hashCode

Hver er meðaltals tími fyrir leit í BST? $O(\log(N))$

Hvaða röðunaraðferð líkist BST? Quicksort

Hver er versta dýpt á llrbt? $O(\log(N))$ (sama með N hnúta)

Hver eru tengslin á milli 2-3 trjáa og

vinstri hjátrauð-svartra trjáa (lrbt)?
2-hnútar eru hnútar, 3-hnútar eru 3
hnúta tré með rauðan legg til vinstri
Hver er fjöldi aðgerða sem þarf að
framkvæma fyrir vinstri snúning í BST
tré með N hnútum? O(1)
Hve stór er in síða (page) á disk? 4k
Hver er sóknartími til að sækja bæti úr
minni tölvu (RAM)? 100ns
Hvernig þurfa equals og hashCode að-
ferðirnar að passa saman? hashCode þarf
að skila sama gildi fyrir tvo hluti ef þau
eru equals
Hver er tíminn sem það tekur að reikna
út hashCode fyrir String hlut af lengd N?
O(N)
Hver er meðaltalstími fyrir leit í hak-
katöflu með tengdum listum? O(1)
Hver er ókostur við að geyma netið sem
tvívítt 0-1 fylki? O(N²) minnisnotkun
Hvaða gagnagrind notar BFS við leit?
Biðröð (FIFO, queue)(sama í stefndu
neti)
Hvaða gagnagrind notar DFS við leit?
Stafla eða fallaköll
Hve langan tíma tekur BFS að skoða allt
netið? O(E+V)
Hvaða reiknirit er hægt að nota til að
finna samhengispætti neta? Union Find,
BFS, DFS
Hvaða reiknirit er hægt að nota til að
finna hvort net hafi hring? DFS og BFS
Hvaða grunnleitaraðferð notar grann-
fræðileg röðun (topological sort)? DFS
Hvaða skilyrði eru fyrir því að hægt sé
að finna grannfræðilega röðun í stefndu
neti? Netið þarf að vera laust við stefnda
hringi, þarf að vera DAG, Directed
Hvaða leitarreiknirit gefur stystu leið í
stefndum netum? BSF
Hvað er spannandi tré (spanning tree)?
Hlutnet sem er samhangandi, án
hringja (tré) og inniheldur alla hnúta
netsins
Hvaða gagnagrind nota reiknirit Prims
til að halda utan um leggi sem gætu bæst
við MST? Forgangsbiðröð með vísi (e.
Indexed Priority Queue)
Í hvaða röð eru hnútar skoðaðir
í reikniritinu fyrir stystu leið í
DAG?grannfræðilegri röð (topologi-
cal order)
Hvernig er lengsta leið fundin í
DAG?stysta leið notuð með neikvæðar

vigtr
Hver er tímakeyrslan á stystu leið í
DAG?E+V
Hvaða netareiknirit er notað til að
finna út hvort brigðgeng stöðuvél
(NFA) samþykki streng?DFS, BFS eða
reachability
Hver er munurinn á löggengri stöðuvél
(DFA) og brigðgengri stöðuvél (NFA)?
brigðgeng stöðuvél hefur margar leggi
með sömu tákn úr sama ástandi.
Dæmi
4 3 2 1 0 9 8 7 6 5 push fyrstu 5 tölunum,
þá prentast 4 3 2 1 0. aftur fyrir seinni 5
tölurnar og þá prentast út 9 8 7 6 5. Þessi
röð er því möguleg.
public class Quick public static
void sort(Comparable[] a) StdRan-
dom.shuffle(a); sort(a, 0, a.length - 1);
private static void sort(Comparable[]
a, int lo, int hi) if (hi <= lo) return; int
j = partition(a, lo, hi); sort(a, lo, j-1);
sort(a, j+1, hi); assert isSorted(a, lo, hi);
private static int partition(Comparable[]
a, int lo, int hi) int i = lo; int j = hi + 1;
Comparable v = a[lo]; while (true) while
(less(a[++i], v))if (i == hi) break; while
(less(v, a[--j]))if (j == lo) break; if (i >= j)
break; exch(a, i, j); exch(a, lo, j); return j;

public class int binarySearch(int arr[],
int l, int r, int x) if (r >= l) int mid = l +
(r - l) / 2; if (arr[mid] == x) return mid; if
(arr[mid] > x) return binarySearch(arr, l,
mid - 1, x); return binarySearch(arr, mid
+ 1, r, x); return -1; int main(void) int
arr[] = 2, 3, 4, 10, 40 ; int n = sizeof(arr)
/ sizeof(arr[0]); int x = 10; int result
= binarySearch(arr, 0, n - 1, x); (result
== -1) ? printf(“Element is not present
in array”): printf(“Element is present at
index return 0;

public class Graph private int V; private
List<int>[] adj; Graph(int v) V = v; adj
= new List<int>[v]; for (int i = 0; i <
v; ++i) adj[i] = new List<int>(); void
AddEdge(int v, int w) adj[v].Add(w); void
DFSUtil(int v, bool[] visited) visited[v]
= true; Console.WriteLine(v +); List<int>
vList = adj[v]; foreach(var n in vList)
if (!visited[n]) DFSUtil(n, visited); void
DFS(int v) bool[] visited = new bool[V];
DFSUtil(v, visited); public static void

```
Main(String[] args){Graph g = new  
Graph(4); g.AddEdge(0, 1); g.AddEdge(0,  
2); g.AddEdge(1, 2); g.AddEdge(2, 0);  
g.AddEdge(2, 3); g.AddEdge(3, 3); Con-  
sole.WriteLine("Following is Depth First  
Traversal "+ "(starting from vertex 2)");  
g.DFS(2); Console.ReadKey();  
  
public class DijkstraSP  
private double[] distTo;  
private DirectedEdge[] edgeTo;  
private IndexMinPQ<Double> pq;  
public DijkstraSP(EdgeWeightedDigraph  
G, int s)  
for (DirectedEdge e : G.edges())  
if (e.weight() < 0)  
throw new IllegalArgumentException(“edge "+ e + "has negative weight");  
distTo = new double[G.V()];  
edgeTo = new DirectedEdge[G.V()];validateVertex(s);  
for (int v = 0; v < G.V(); v++)  
distTo[v] = Double.POSITIVE_INFINITY;distTo[s] =  
0.0;  
pq = new IndexMinPQ < Double >  
(G.V());  
pq.insert(s, distTo[s]);while(!pq.isEmpty())  
intv = pq.delMin();for(DirectedEdge e :  
G.adj(v))  
relax(e);assertcheck(G,s);  
privatevoidrelax(DirectedEdgee)  
intv = e.from(),w = e.to();  
if(distTo[w] > distTo[v] + e.weight())  
distTo[w] = distTo[v] + e.weight();  
edgeTo[w] = e;if(pq.contains(w))  
pq.decreaseKey(w, distTo[w]);  
elsepq.insert(w, distTo[w]);  
  
publicclassBST <  
KeyextendsComparable < Key >  
, Value >  
privateNoderoot;privateclassNode;  
privateKeykey;privateValueval;  
privateNodeleft, right;privateintsize;  
publicNode(Keykey, Valueval, intsize) <  
this.key = key;this.val = val;this.size =  
size;  
publicBST()publicbooleanisEmpty()  
returnsize() == 0;publicintsize()  
returnsize(root);privateintsize(Nodex)  
if(x == null)return0;elsereturnx.size;  
publicbooleancontains(Keykey)  
if(key == null)thrownew  
IllegalArgumentException(“argumentto  
contains()isnull”);
```

```
returnget(key)!  
null;publicValueget(Keykey)  
returnget(root, key);privateValueget  
(Nodex, Keykey)  
if(key == null)thrownew  
IllegalArgumentException(“callsget()withanu  
key”);  
if(x == null)returnnull;  
intcmp = key.compareTo(x.key);  
if(cmp < 0)returnget(x.left, key);  
elseif(cmp > 0)returnget(x.right, key);  
elsereturnx.val;
```

Sort	best	worst
Selection	$O(N^2)$	$O(N^2)$
insertion	$O(N)$	$O(N^2)$
quick	$O(N\log(N))$	$O(N^2)$
merge	$O(N\log(N))$	$O(N\log(N))$
heap	$O(N\log(N))$	$O(N\log(N))$
Problem	Algorithm	Time

Problem	Algorithm	Time	Space
path	DFS	E+V	V
Shortest path	BFS	E+V	V
Cycle	DFS	E+V	V
Directed path	DFS	E+V	V
Directed Cycle	DFS	E+V	V
Topological sort	DFS	E+V	V
Connected components	DFS	E+V	V
Minnimum spanning tree	Kruskal	E log E	E + V
Minnimum spanning tree	Prim	E log V	V
Shortest paths(nonneg weights	Dijkstra	E log V	V
Shortest paths(no cycles)	Topological sort	E+V	V

