Analysis of Algorithms CS323

April 13. 2016

Notes by: Xiaoxiao Wang

Homework 7 solution:

1.

	Binary Heap	**Fib Heap**
FindMin	O(1)	O(1)
RestoreHeap	O(logn)	O(m)
MergeHeap	O(n)	O(1)
Insert	O(logn)	O(1)
Structure	One tree	Forest (collection of trees)

2.

	2	**3**	**4**	**5**	**6**
1	7(1)	9(1)	∞(1)	∞(1)	14(1)
2	-	9(1)	22(2)	∞(1)	14(1)
3	-	-	20(3)	∞(1)	11(3)
6	-	-	20(3)	20(6)	-
4	-	-	-	20(6)	-
5	-	-	-	-	-

*distance(pred)

3.

	2	**3**	**4**	**5**	**6**
1	7(1)	9(1)	∞(1)	∞(1)	14(1)
2	-	9(1)	15(2)	∞(1)	14(1)
3	-	-	11(3)	∞(1)	2(3)
6	-	-	11(3)	9(6)	-
5	-	-	11(3)	-	-
4	-	-	-	-	-

*distance(pred)

Total cost: 33

Total edge: 5

4. a) Number of pairs: n^2

Number of operations: n

$\downarrow$

O(n^3)

b) maxEndingHere = a [1];

maxSoFar = a [1];

for 1 = 2 to n

maxEndingHere = max (maxEndingHere + a [i], a[i]);

maxSoFar = max (maxSoFar, MaxEndingHere);

return maxSoFar;

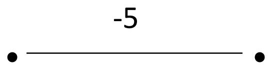
Lecture

Belman-Ford SSSP Algorithm

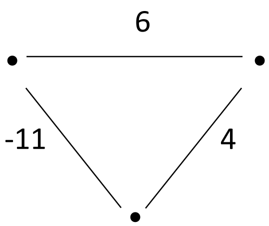
Accommodates negative weights edge (Dijkstra doesn’t work with negative weights)

```
//c: cost   s: source   d: distance   p: pred

for v∈ V

    d [v] = c [s,v]

    p [v] = s

for each remaining vertices (v – {s})

    for each edge (u, v)

        if d [v] + c [u, v] < d [v]

            d [v] = d [u] + c [u, v]

            p [v] = u

        end if

    end for

end for

for each edge (u, v)

    if d [u] + c [u, v] < d[v]

        return NegativeWeightCycle
```

negative weight edge:



negative weight cycle:



total weight: -1

Kruskal's MST Algorithm

"disjoint set operations"

(n sets) "Make-set" V_1 V_2 V_3 V_4 ... V_n

Sort edges in nondecreasing order

depends on the range
comparison or noncomparison sorted

"Find"

"Union" if Find(U) ≠ Find(V)

Union(U,V)

eg. 2-4 V_2 V_4

1 2 3 4 ... n

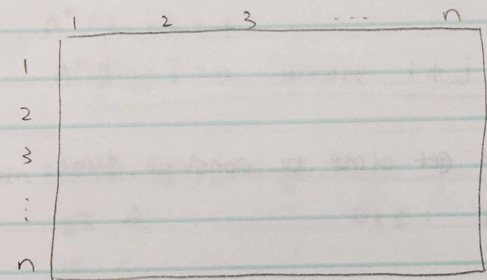
"Make set" | 1 | 2 | 3 | 4 | ... | n | — set representity

1-d: Find(2) Find(4) Find-Union

Find: Find the representity . they connect to each other but have different representity

Union: change the representity to the same

another way: 2-d array.



if same component, they are connected.

1-d: n Finds , n-1 union $O(n^2)$

anothe approach : make trees

1 operation to combine the tree.

go up to the root, find the representity.

Smallest : 1 operation largest : n-1

Worst : n-1 Find $O(n^2)$

Compressing Find (C-Find) when find the ~~path~~, compress

Weighted Union (W-Union)

if do 1 ; $n \log n$

do both : close to constant time.

↓

(not exactly)

inverse of Ackermon's function

$$A(m,n) = \begin{cases} n+1 & \text{if } m=0 \\ A(m-1,1) & \text{if } m>0 \text{ and } n=0 \\ A(m-1, A(m,n-1)) & \text{if } m>0 \text{ and } n>0 \end{cases}$$

19*

$\lg^*$

$\lg n$ $\lg 256 = 8$

$2^8 = 256$

$2^2 = 4$ $2^{2^2} = 2^4 = 16$ $2^{2^{2^2}} = 2^{16} = 65536$

$2^{2^{2^{2^2}}} = 2^{65536}$

even large #, still get close to constant time.

$n\alpha(n, m)$

Floyd's Algorithm (APSP)

Worshalls

for $k=1$ to n

for $i=1$ to n

for $j=1$ to n

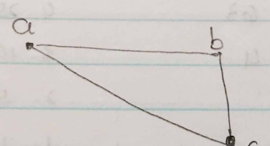
$T[i,j] = (T[i,k]$ and $T[k,j])$ or $T[i,j]$

$\Theta(n^3)$

* if there is path from i to k and path k to j then path i to j.

$a=b$ and $b=c$ then $a=c$

$a<b$, $b<c$ $\Rightarrow$ $a<c$



$a \to b$, $b \to c$

then $a \to c$

transtive closure connectivity.

$T[i,j] = \begin{cases} 1 & \text{if path from } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$

Adjacent Matrix and raise to the 1st power

A^1 --

A^2 path of length 2

A^3 --

$\vdots$ --

A^{n-1} -- $\sum_{i=0}^{n-1} A^i$

A^0 = identity matrix, each node is connected to itself.

$A^0[i,i] = 1$

$A^0[i,j] = 0$ where $i \neq j$

Coin-Making-change

e.g.	25¢	63¢:	2 × 25¢
	10¢		1 × 10¢ ← 0 × 5¢
	5¢		3 × 1¢
	1¢		

What if:

4¢	6¢:	4¢ + 1¢ + 1¢ (3 coins)
3¢		3¢ + 3¢ (2 coins)
1¢		

c: # of coins.

index

Value

$$c(i,j) = \min(1 + \underline{c(i - v[j], j)}, c(i,j))$$

amount of change we need to produce

i: 63

j: 4

4. 25¢
3. 10¢
2. 5¢
1. 1¢

$(\underbrace{63¢ - 25¢}_{38¢}, 4)$

$c(i,1) = i$

$c(i,2) =$

$$\boxed{C(n,r) = C(n-1,r) + C(n-1,r-1)}$$

$C(n,0) = 1 \quad C(n,1) = n$