Intro to Algorithms - HW5 Solutions

- Pranay Anchuri (GTA)

4.13 a) To determine if there is a route b/w s & t satisfying the given conditions, first remove any edge $e \in E$ for which $l_e > L$ (strictly greater).

Now, run BFS (DFS) starting from either 's' or 't' and see if you can reach 't' or 's' respectively.

Complexity: $O(|V| + |E|)$

4.19) Modify the given graph as follows:

for any edge $e = (x, y) \in E$,

$l(x,y) \longrightarrow l(x,y) + cost[y]$

i.e add the cost of endpoint.

→ Run dijkstra on the modified graph (ignore vertex costs)

→ min cost (s, u) in original graph = dijkstra cost (s, u) in modified graph + cost(s) cost on source vertex

4.21

a) i) Construct a graph as follows:

→ A node for each currency

→ Edge between every pair of currencies.

→ $wt(c_i, c_j) = -\log(r_{ij})$

c_i: i^{th} currency, c_j: j^{th}, r_{ij}: exchange

ii) Run all-pairs shortest path

iii) Return the pair (u, v) with shortest path.

4) CLRS - 24.3.6

→ Run Dijkstra with $\log\left(\frac{1}{r(u,v)}\right)$ as the weight on the edge between u and v.

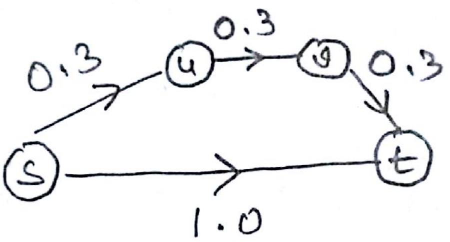
→ Shortest path in the modified graph corresponds to most reliable path.

$\Rightarrow$ Note that $\log\left(\frac{1}{r(u,v)}\right)$ is ≥ 0

5.5

a) Spanning tree doesn't change because the edges remain sorted.

b) Shortest path can change.



Before:
$s \to u \to v \to t$

After:
$s \to t$.

5.7

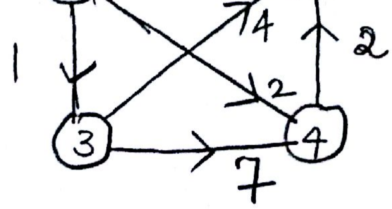
a) Sort the edges in non-increasing order.

b) Run Kruskal algorithm.

5.8

a) Directed graph: It is possible not to share any edges.

Example:



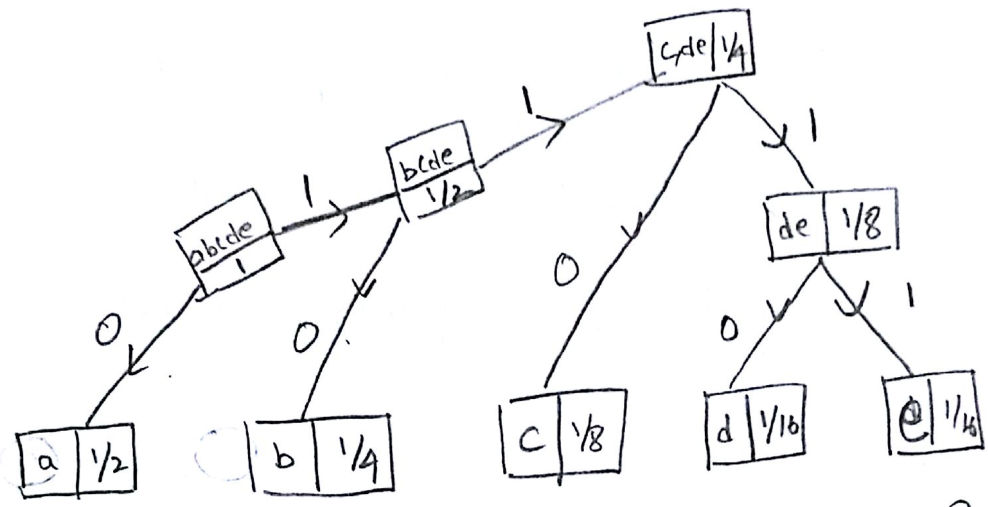
MST contains (1,3), (1,4) (4,2)

Shortest path tree from 3 uses (3,2) (3,4) (2,1)

b) Undirected graph: NOT possible.
Minimum cut edge is common b/w MST and SPT.

5.14

a)



a : 0   b : 10   c : 110   d : 1110
e : 1111

b) 1 M characters in file

Character	# in file	# bits	Total
a	5,00,000	1	5×10^5
b	250 K	2	5×10^5
c	125 K	3	3.75×10^5
d	62.5 K	4	2.5×10^5
e	62.5 K	4	2.5×10^5
			18.75×10^5

$= 1.875 \times 10^6$

5.20 Perfect matching:

Repeat the following procedure until the ~~graph~~ Tree is empty.

a) Pick a leaf node (i.e node w/o children and 1 parent)

↳ This edge is present in the output.

b) Remove above selected edge and the other outgoing edges from that parent node

c) Go to $\boxed{a}$

5.21 Feedback edge set.

a) Negate the edge weights → G

b) Run spanning tree algorithm

c) Return: E - spanningTreeEdges(G)

Complexity: $O(|E| \log |E|)$

Logic :→ Running spanning tree with -ve weights chooses edges with large (actual) weight.

→ Moreover, spanning tree algorithm removes edges that create a cycle. In this case we want such edges.

6.3 Can be solved in linear time as the distances are sorted.

Use the following recurrence relation.

$$f(i) = \begin{cases} P_1 & \text{if } i = 1 \\ \max(f(i-1), f(p(i)) + P_i) & \text{o/w.} \end{cases}$$

the $p(i)$ is the predecessor of i^{th} location.

$$p(i) = \max \{ j \mid j \le i, m_i - m_j \le k \}$$

$f(i)$ is the profit using first 'i' locations.

6.5

a) 1 pebble: 4 pattern (4 rows)

2 pebble: 3 patterns 13, 14 | 24

0 pebble: 1 pattern

8

b) Let $P_1, P_2 \dots P_8$ denote 8 possible patterns.

$C_{P_j}[i]$ = value of squares covered by P_j th pattern in column i

$$C_{P_j}[i] = \max\left\{ C_{P_{j'}}[i-1] + \text{value of } P_j \text{ in column } i \right\}$$

$P_{j'}$ are all other compatible pattern for previous column.

→ Return $\max_{1 \le j \le 8} \{ C_{P_j}[n] \}$.

6.17

$$c[i] = \begin{cases} \text{True} & \text{if it is possible to change } `i' \\ \text{False} & \text{Otherwise} \end{cases}$$

→ We need $c[v]$

Recurrence: $c[0] = \text{True}$

$$c[u] = \begin{cases} \text{is any true } (c[u - x_j],\ x_j \leq u) \\ \text{False} & \text{otherwise} \end{cases}$$

$0 < u \leq v$

Time: $O(n \times v)$