
G
O
D
O
O
O
D

Section 1: Greedy Algorithms and Minimum Spanning Trees

Question 1: Consider the following version of Prims algorithm

Input: $G(V, E)$
Output: A minimum spanning tree T of G
Initialize a priority queue Q of V nodes
for all $v \in V$:
  $v.\text{key} = \infty$
  $v.\text{parent} = \text{NIL}$

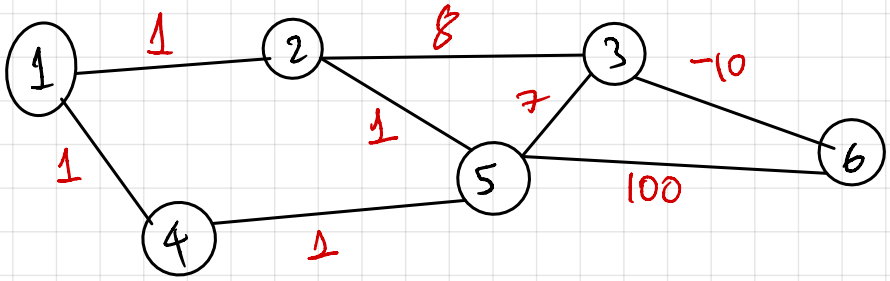
$v_0.\text{key} = 0$

for all $v \in V$:
  insert(Q, v)

while $Q \neq$ empty
  $u =$ **Extract-Min**(Q)
  for all $v \in$ **Adj**$[u], v \in Q$
    if $v.\text{key} >$ **weight**$(e(u, v))$:
      $v.\text{parent} = u$
      $v.\text{key} =$ **weight**$(e(u, v))$ using **Decrease-Key** operation on Q
return MST T obtained by tracing the parent pointers

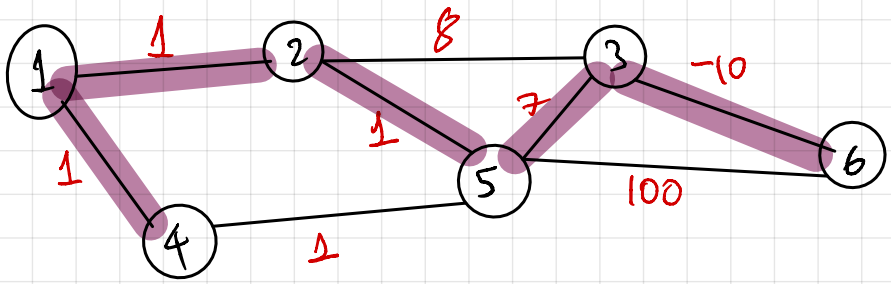
Q.1.a Analyze the number of times the inner for loop is run during the execution of the algorithm and prove that it is O(E)? **(2 points)**

Q.1.b Run Kruskal's algorithm on the following graph and report the order in which the edges are selected into the minimum spanning tree? (**4 points**)



Q.1.a The inner for-loop look at a vertex v which is adjacent to u, where u is the currently extracted min weight vertex from the priority queue Q. Since a vertex u is extracted from Q exactly once for every edge u–v in the graph, the for loop considers this edge when either of u, v is extracted from the queue. Thus every edge in G is inspected at most once by the for-loop during the execution of the Prims algorithm. Hence the time complexity is O(E), where E is the number of edges.

Q.1.b



In your answer choose any that works; could have also chosen (1,4), (4,5) or (2,5) instead

Edges chosen by Kruskals algorithm (in order): 3–6 (weight: −10), 1–2 (w: 1), 2–5 (w: 1), 1–4 (wt: 1), 5–3 (w: 7)

(when there are multiple safe next edges you may choose any)

Q.1.c Given an undirected connected graph G, prove by induction that the during the execution of Kruskal's algorithm, the set of edges already selected by the algorithm belongs to some minimum spanning tree T of G. **(5 points)**

You may use the following induction hypothesis:

Induction Hypothesis: Let let e_1,e_2,..,e_{k-1} be the set first k edge selected by the Kruskals algorithm. Then there is a minimum spanning tree T of G which contains all the of the edges e_1,e_2,…,e_{k-1}.
Hint : In a minimum spanning tree there are n vertices and n-1 edges, thus k is between 1 and n-1.

Question 2 Solve the recurrence relation : T(n) = T(3n/4) + O(n) and explain your reasoning and steps. **(5 points)**

Hint : This is the recurrence relation for the running time of randomized quick select, if every pivot that you chose is a 3/4th approximate median. You could solve this by analyzing the recurrence tree for this recurrence relation and compute the cost of all the nodes at a certain level and sum them all up. You can use any other method that you know as well.

$T(n) = T(3n/4) + cn$. The recurrence tree looks like

$T(n) \rightarrow$ cost $= cn$

$T(3n/4) \rightarrow$ cost $c(\frac{3n}{4})$

$T(\frac{3^k}{4^k} n) \rightarrow$ cost $c(\frac{3^k}{4^k} n)$

The tree has height k, where $\frac{3^k}{4^k} n = 1$ (when input size becomes 1, recursion stops)

solving for k

$(\frac{3}{4})^k n = 1 \Rightarrow n = (\frac{4}{3})^k$

$\Rightarrow \log n = k \log 4/3 \Rightarrow k = \frac{\log n}{\log 4/3}$

Total cost

$$T(n) = \sum_{k=0}^{\log n / \log 4/3} c\left(\left(\frac{3}{4}\right)^k n\right)$$

$$= cn \sum_{k=0}^{\frac{\log n}{\log 4/3}} \left(\frac{3}{4}\right)^k$$

$$\leq cn \sum_{k=0}^{\infty} \left(\frac{3}{4}\right)^k.$$

(Geometric Progression)

The sum of an infinite G.P. with starting term $a = 1 = (3/4)^0$, and common ratio $0 < r = 3/4 < 1$ is $\frac{a}{1-r}$

$$= c \cdot n \times \frac{1}{1 - 3/4} = 4 \cdot c \cdot n = O(n)$$

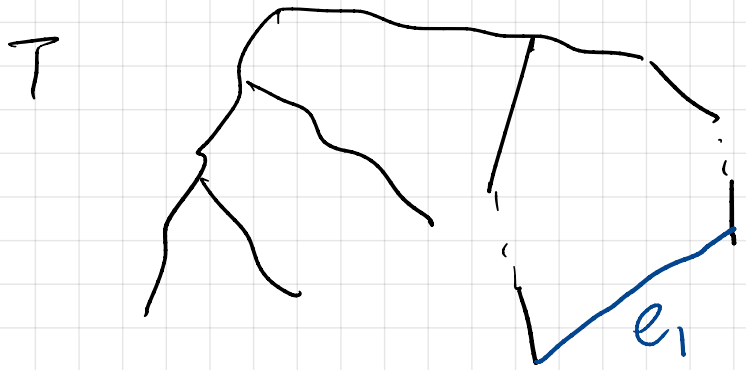
Q.1.c

I.H: Let $e_1 \dots e_k$ be the first k edges chosen by Kruskals algorithm. Then there exists an MST, T, of G that contains all these edges.

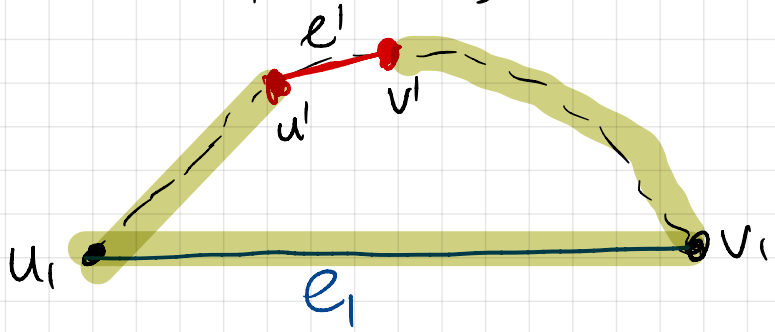
Base case: $k=1$, let $e_1(u_1, v_1)$ be the first edge chosen by Kruskals. At the beginning the algorithm chooses the smallest weight edge in the graph. That is for any $e \in E(G)$

$$wt(e) \leq wt(e_1)$$

Let T be any MST of G. If T contains e_1 then we have proved the induction hypothesis. If not we will add e_1 to T



Since T is a tree which avoids $e_1(u_1, v_1)$, there must be a path in T connecting u_1 to v_1 (since T is a spanning tree it has to contain u_1 and v_1). Thus adding e_1 creates a cycle in T.



Choose any other edge $e'(u', v')$ on this path. Remove it from T.

Thus

$$T^* = T \setminus \{e'(u', v')\} \cup \{e_1(u_1, v_1)\}$$ is a tree

as we removed the only cycle in T created by adding e_1 and since we did not affect the connectivity of the vertices by removing e'. [If a path connecting two vertices used e', we can now use e_1]

$$\text{weight}(T^*) = \text{weight}(T) - \text{weight}(e') + \text{weight}(e_1)$$

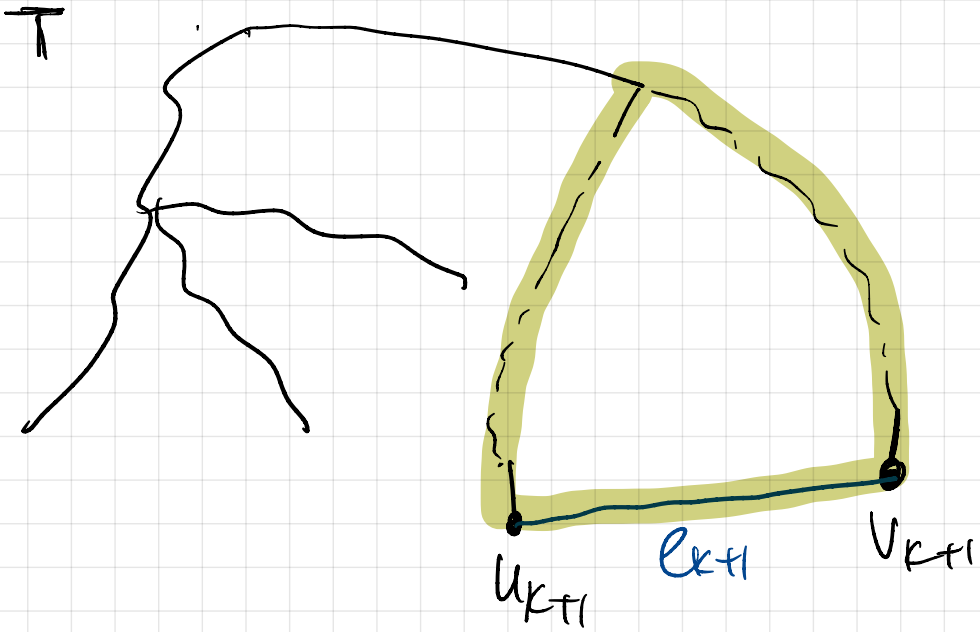
Since $\text{weight}(e') \leq \text{weight}(e_1)$, $\text{weight}(T^*) \leq \text{weight}(T)$

and T^* is an MST containing e_1. This proves Induction Hypothesis

Induction step Let e_{k+1} be the edge added after k^{th} step.

By I.H., there exists an MST T containing $e_1, \ldots, e_k$.

If T also contains e_{k+1} we are done. Otherwise we will add e_{k+1} to T as before.



This creates a cycle. We would like to show that we can remove a an edge $e'(u', v')$ from this cycle which is not any of the edges $e_1 \ldots e_k$.

We first note that not all the edges in the graph can be formed

by a subset of $\{e_1, \ldots, e_k\}$ along with e_{k+1}. If this was the case then e_{k+1} forms a cycle with the previously chosen edges and Kruskal's algorithm would not chose e_{k+1} as the next edge.

Thus in the highlighted cycle, there exists an edge $e'(u', v')$ outside $(e_1, \ldots, e_k)$. Note that since $e'(u', v')$ was already present in the tree T along with edges $e_1, \ldots, e_k$, it is an edge that does not form a cycle with previously chosen edges. Thus e' was also a safe choice for the $k+1^{th}$ edge. The reason Kruskals chose e_{k+1} over e', is because $wt(e') \geq wt(e_{k+1})$

Thus T^* obtained by adding e_{k+1} and removing e' is an MST containing $e_1, \ldots, e_k, e_{k+1}$ (thus proving induction hypothesis) as

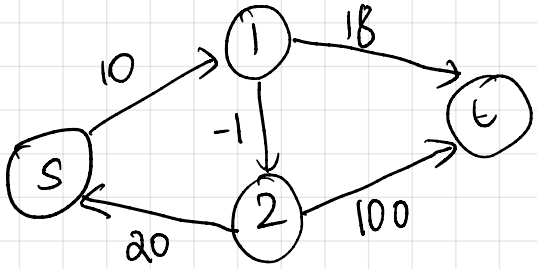
$$wt(T^*) = wt(T) - wt(e') + wt(e_{k+1}) \leq wt(T).$$

Section 2 : Dynamic Programming and Shortest Path algorithms

Question 3: Write the pseudocode for computing U_n = 2 U_{n-1} + 3 U_{n-2} for n > 2, where U_2 = 20, U_1 = 5 using dynamic programming with memoization. Furthermore describe the subproblems, and give an (asymptotic) estimate of the number of subproblems. **(2 points)**

Question 4: Consider the following formulation of the shortest path DP formulation. Given a weighted directed graph G(V,E) with no **negative weight cycles**, let delta_k(u,v) denote the length of the shortest path from u to v that uses at most k edges (and if there is no such path is equal to infinity, and delta_k(u,u) = 0). We can write delta_k(u,v) = min{ delta_{k-1}(u,w) + weight(w->v) ; where w->v is an edge in G}. This DP formulation exploits the optimal substructure property to say that if the shortest path from u to v using at most k edges has w as the last vertex before v, then the sub-path from u to w, must be the shortest path from u to w using at most k-1 edges. Using this formulation, we can compute the shortest path between any two vertices u, v denoted by delta(u,v) by computing delta_n(u,v) as any two vertices which are connected in an n vertex graph is connected by a path of length at most n.

Q.4.a For the following graph, list all the subproblems for computing the shortest path from vertex s to t. Then use the dynamic programming formulation above to compute the length of the shortest path from s to t? **(4 points)**



Q.4.b Write the dynamic programming with memoization that computes delta(s,t) for any two given nodes s and t of a weighted directed graph G using the above dynamic programming formulation. **(5 points)**

Q.4.c Argue that the subproblem graph of the dynamic program formulation above has no directed cycles (that is prove that the subproblem graph is a DAG - Directed Acyclic Graph)?
Hint : The subproblems needed to solve delta_**k**(u,v) are of the form delta_{**k-1**}(u,v). Use this to argue that subproblem graph is a

Q.3. $U_n = 2U_{n-1} + 3U_{n-2}$ for $n>2$ and $U_2=20$, $U_1=5$.

pseudocode: let U be a global integer array containing n elements, initialized to -1

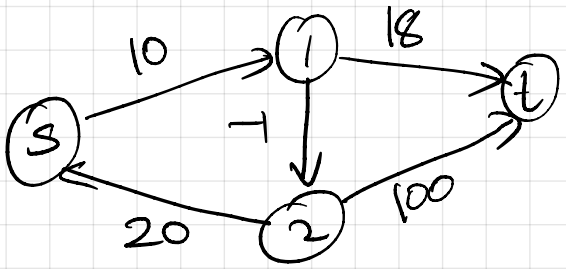
```
compute U(n):
    if n=1
        U[1] = 5
        return
    if n=2
        U[2] = 20
        return
    else
        if (U[n-1] = -1) : compute U(n-1), if (U[n-2]=-1): compute U(n-2)
        U[n] = 2xU[n-1] + 3U[n-2]
```

subproblem: Each subproble is U_i for $1 \le i \le n$

#-of subproblems $O(n)$

Question 4

$$\delta_k(u,v) = \min_{w \text{ s.t. } (w \to v) \in E(G)} \{ \delta_{k-1}(u,w) + \text{weight}(w \to v) \}$$



To compute the length of the shortest path from s to t, $\delta(s,t)$ we can instead compute $\delta_k(s,t)$ where $k=3$ as any path in the given graph can at most 3 edges (as there are only 4 vertices)

[Note: you will not be penalized for choosing a larger value of k, like $k=4$ (# of vertices), $k=5$ (# of edges) etc]

Subproblems for computing $\delta_3(s,t)$

$$\delta_3(s,t) = \min \{ \delta_2(s,1) + \text{weight}(1 \to t), \delta_2(s,2) + \text{weight}(2 \to t) \}$$
$10 + 18 = 28$ $\quad$ $9 + 100 = 109$

$$\delta_2(s,1) = \min \{ \delta_1(s,s) + \text{weight}(s \to 1) \}, \quad \delta_1(s,s) = 0$$
$0 + 10 = 10$

$$\delta_2(s,2) = \min \{ \delta_1(s,1) + \text{weight}(1 \to 2) \}$$
$10 + -1 = 9$

$$\delta_1(s,1) = \min \{ \delta_0(s,s) + \text{weight}(s \to 1) \}, \quad \delta_0(s,s) = 0$$
$0 + 10 = 10$

Thus, length of $\delta(s,t) = \delta_3(s,t) = 28$