Homework3

Qinyun Song

1 Uniqueness of spanning trees

1. *Proof.* when using the *Kruskal* algorithm to get the MST of G, one first need to sort the edges by their weights from small to large. If there are two different MST of G, then at some point, for an edge e, one MST chooses this one but the other one disgards this edge. However, by using the correctness of *Kruskal* algorithm, if the edge e is the edge satisfing the requirement of the algorithm, it must be selected. Otherwise the resulting tree is not MST of the original graph. Thus by contradiction, we can proof that there can only be one MST. $\square$

2. *Proof.* By using the *Kruskal* algorithm, the edges first sorted by their weights from small to large. Then if there are two different MST of G, there exist an edge e_i such that the decisions before it are the same for the both MST but one MST chooses edge e_i while the other doesn't. Adding this edge e_i into the MST that doesn't choose it. Then there must be a cycle consisting this edge. The edge e_i should not be the edge with maximum weights in the cycle otherwise both MST will not select this edge. Then since the weights involved in one cycle are all distinct, we can remove the edge with maximum weight in the cycle and then get a tree with smaller total weights. This is contradict with the assumption that the second tree is also a MST. So there should be only one MST of the graph. $\square$

2 Reverse greedy

Claim: The process will result in an MST and the produced MST is the same as the MST produced by *Kruskal* algorithm.

Proof. Proof this claim by induction. Here, suppose the tree produced by reverse greedy is T and the MST produced by *Kruskal* algorithm is R. Suppose the edges are sorted by their weights from large to small.
For the most basic situation, there are no edges added to the tree so $T = R$.
Suppose for edge $e_1, \cdots, e_{i-1}$, the choices of both algorithm are the same. Then for the current edge e_i, if it is not chosen by T, it means that there will be other edges with smaller weights connecting all the vertices. Then it will not be chosen by R too. Because when it is considered by R, the two vertices it connects are already connected. On the other hand, if the edge is chosen by T, then it shows that there are no other edges with less weights connecting the two vertices it connects. So it must be chosen by R, because when R enumerates the edge e_i, the two vertices it connected is still not connected yet. So R must choose this edge.
From above, we can see that, the edge T chooses, it will also be in R. And the edge T disgards, it will not in R. Thus the two algorithm will produce the same tree.
Since the tree produced by *Kruskal* is MST and the tree produced by reverse greedy is the same as the MST, the tree produced by the reverse greedy is also a MST. $\square$

3 Weary traveler

Solve this problem by construct a graph G. First of all, all the cities are numbered from 1 to n. For each city i, add two vertices v_{i_1}, v_{i_2} to G. And then for each flight from city i to j, add an edge from v_{i_2} to v_{j_1} and its length equals the time needed for the flight. Finally for each city i, add edge from v_{i_1} to v_{i_2} with length equals 10. After that, we can get a new graph $G = (V, E)$ where $|V| = 2n$ and $|E| = n + m$. For each city i, the vertex v_{i_1} means arriving at i and the other vertex v_{i_2} means leaving the city i. For each edge added according to each flight, it shows the time needed to take the flight from the start to the destination city. For edge edge from v_{i_1} to v_{i_2}, it means that it takes 10 minutes to take the next flight.
Now suppose the traveler wants to travel from city i to j, using the *Dijkstra* algorithm, calculate the minimum distance between node v_{i_2} and v_{j_1}. This is the minimum time it costs.
The time complexity is $O((m + n) + 2nlog(2n)) = O(m + n + nlogn)$.

4 Shortest path distances

Proof. Suppose there exist three points u, v, w such that

$$d(u, v) + d(v, w) \leq d(u, w)$$

the distance from u to w can be reduced by going from u to v and finally to w. By doing so, the minimum distance between u and w is reduced to the sum of $d(u, v)$ and $d(v, w)$. Since the sum is less than the original $d(u, w)$, it is a contradiction that $d(u, w)$ is supposed to be the minimum length between u and w. Thus there cannot exist any three points satisfying the above equation. So for any three vertices u, v, w, we must have

$$d(u, v) + d(v, w) \geq d(u, w)$$

□

5 Approximating distances

1. *Proof.* Suppose there exist two vertices u and v, their minimum distance is $d(u, v)$. And the algorithm chose one vertice s_i and returns the distance $|d(v, s) - d(u, s)|$. And assume that the returned distance is bigger than $d(u, v)$. On the one hand, if $d(v, s) \geq d(u, s)$, then we can have

$$d(v, s) - d(u, s) \geq d(v, u)$$

The equation can be changed into

$$d(s, v) \geq d(s, u) + d(u, v)$$

This is contradiction with the correctness of *Dijkstra* algorithm. Because the distance between s and v can be further reduced which means that $d(s, v)$ is not the minimum distance between them. This is wrong according to *Dijkstra* algorithm.
One the other hand, if $d(v, s) \leq d(u, s)$ we can have

$$d(u, s) - d(v, s) \geq d(v, u)$$

By doing some manipulation, we can have

$$d(s, u) \geq d(s, v) + d(v, u)$$

This also shows that $d(s,u)$ is not the minimum distance between s and u. Thus violate the correctness of *Dijkstra* Algorithm.
So After all, the distance returned by the algorithm will be no larger than the real minimum distance. $\square$

2. The distribution of the probability is binomial distribution. So with each vertex whose probability is $100/n$, the expected number of vertices selected is

$$E = n \times 100/n = 100$$

Since for each vertex chosen, the algorithm will run the *Dijstra* algorithm once, so the expected running time should be $O(100 \times (m + nlogn)) = O(m + nlogn)$.

3. *Proof.* For u, v with $d(u,v) = \alpha n$, consider the chosen point s. Suppose $d(u,s) = x$ and $d(v,s) = y$. Then the return value would be

$$d(u,v) - x - y$$

Since we want to calculate the situation when return value is less than $(\alpha - 0.1)n$, we can say that

$$d(u,v) - x - y = \alpha n - x - y \leq (\alpha - 0.1)n$$

So we want that

$$x + y \leq 0.1n$$

So the probability that the returned value less than $(\alpha - 0.1)n$ equals the situation that the chosen s cannot satisfy the above equation. The probability of that situation would be

$$(1 - \frac{100}{n})^x \times (1 - \frac{100}{n})^y = (1 - \frac{100}{n})^{x+y} \leq (1 - \frac{100}{n})^{0.1n} \leq e^{-10} \leq 0.001$$

So the probability that the return value less than $(\alpha - 1)n$ is less than 0.001. $\square$

6 Line Layout

Proof. For the two-dimensional grid G with $n = a^2$, suppose it has a layout f. Then there exist a point s with $f(s) = 1$ and another point e with $f(e) = a^2$. So we know that

$$|f(s) - f(e)| \leq 2a$$

Since the length of the layout is a^2. Then for every point u, v with $d(u,v) = 1$, we can say that

$$|f(s) - f(e)| \leq 2a \leq 2a \times d(u,v)$$

Also, since $d(u,v) = 1$ and $|f(u) - f(v)| \geq 1$, we can have

$$|f(s) - f(e)| \leq 2a \times d(u,v) \leq 2a \times |f(u) - f(v)|$$

By moving the variable a to the same side, we can have

$$|f(u) - f(v)| \geq a/2$$

$\square$