JichenDai_CS600_Final_Exam

1. Modify:

The modification is using an **unordered linked list** to implement priority queue Q.

Since ***Inserting*** a new element costs **O(1)** time, ***reomveMin*** costs **O(n)** time, we perform ***m*** insertions and ***n*** removeMin. The total time is O($m+n^2$).

Additionally, because m is O(n^2), the total running time is **O(n^2).**

2. Algorithm Description:

1. We can implement a maximum spanning tree algorithm by modifying the minimum spanning tree algorithm. This is easy to implement, we just need to multiply each weight by -1.

2. Using this maximum spanning tree algorithm, we can find the maximum capacity path from s to t.

3. Since it's similar to minimum spanning tree algorithm, the time cost is O((n+m)logn).

3. Proof:

First, proof this problem is in NP:

Let the number of nodes in H be ***n,*** a subset of n nodes is G. When a mapping of nodes in H to nodes in G is given, we can check edges in H one by one if it is corresponding to a edge in G. This can be done in polynomial time.

Second, proof this problem is NP-hard:

We can reduce **Hamiltonian Cycle** to this problem.

Given graph G = (V, E) in the Hamiltonian Cycle Problem, Let H be a sub-graph of G and H is a simple cycle of all nodes.

The steps above can be done in polynomial time.

H is sub-graph of G if and only if G is Hamiltonian.

Thus, SUBGRAPH-ISOMPRPHISM is NP_complete.

4. Greedy is a 2-approximation solution for this problem.

There are two ways between two point a and b, greedy strategy will always choose the shorter way. Let X be a router on the shorter way, and Y be a router on the longer way. X and Y won't transmit information to each other.

Let G_X and G_Y denote the number of transmission across X and Y in greedy solution.

Let O_X and O_Y denote the number of transmission across X and Y in optimal solution.

It is obvious that: $G_X + G_Y <= O_X + O_Y$

Let G_X be the larger one, then $G_X <= O_X + O_Y <= 2*\text{Max}\{O_X, O_Y\}$

Since max load, L_O, in optimal solution is $\text{Max}\{O_X, O_Y\}$, $G_X <= 2*L_O$

So, greedy is a 2-appriximate solution.

5. Data Structure: Using linked lists to implement it and each has size of B.

Proof:

When implement enqueue, we take this element to the end of the linked list and check whether there are B elements. If there are B elements, then create a new list and add this element to this new list. If the length is less than B, then just add this element. There will be O(n/B) times of disk transfer.

When implement dequeue, in the worst case, we may have to access all blocks to move elements up to down. This requires O(n/B) times of disk transfer.

So the total time is O(n/B).

6. Answer:

Since a<x<b, if we plot point (a,b) and (x,x) in a two-sided range, (a,b) is at the upper left of (x,x).

In this two-sided range, each interval is a two-dimensional point.

Since two-sided range is a simple case of three-sided range, we can use priority search tree to find all the intervals (*PSTSearch(x1,x2,y1,v) in chapter 21.22)*.

The time cost of priority search tree is O(logn + k).

According to **Lemma 21.3**, **the space usage is O(n)**.

7. Answer: A brute-force algorithm is enough to solve it in O(n^2) time.

A simple polygon don't contain crossed edges. We can check each edges if it cross with the rest of the edges. We will check for O(n^2) times.

Determine whether two lines intersect takes O(1) time, so the total running time is O(n^2)

8. Answer:

We First construct a prefix trie. A prefix trie can be easily modified from a suffix tire mentioned in Section 23.5.3.

Using algorithm prefixTrieMatch(T, P), which is also modified from Algorithm suffixTrieMatch(T, P) in Section 23.5.3, we can find matching pattern.

According to Theorem 23.7, time cost is O(dm), where m is the length of pattern.

Since d = 4, m = O(n), the time cost is O(n).

9. Answer:

1.Standard form:

Maximize: z = x1 + 6 x2

Subject to: x1 <= 200

x2 <=300

x1 + x2 <= 400

x1, x2 >= 0

2.Slack form:

Maximize: $z = x1 + 6\,x2$

Subject to: $x3 = 200 - x1$

$x4 = 300 - x2$

$x5 = 400 - x1 - x2$

$x1, x2, x3, x4, x5 >= 0$

We first keep x1 at 0 and raise x2. Since x4 constrains x2 most, we pivot x4 and x2.

Maximize: $z = 1800 + x1 - 6x4$

Subject to: $x3 = 200 - x1$

$x2 = 300 - x4$

$x5 = 100 - x1 + x4$

$x1 , x2, x3, x4, x5 >= 0$

Now we increase x1, since x5 constrains x1 most, we pivot x5 and x1

Maximize: $z = 1900 - x5 - 5x4$

Subject to: $x3 = 100 - x4 +x5$

$X2 = 300 - x4$

$X1 = 100 - x5 + x4$

$x1 , x2, x3, x4, x5 >= 0$

Now, all variables in objective function is have negative coefficients. So when both x5 ans x4 equal to zero(**x1 = 100, x2 = 300**), we got the optimal solution.

The optimal profit = 100*1 + 300*6 = **1900**