Final Exam CS 600 Spring 2018

Instruction: *Answer the following questions in this document or a word or PDF document and submit it in Canvas according to the Final Exam Procedure.*

1. (12 Points) Consider n space stations that communicate using laser between each pair of stations. Naturally, the energy needed to connect two stations depends on the distance between them. Therefore, each pair of stations has a different known energy requirement. We want to connect all these stations together using the minimum amount of energy possible. Describe an algorithm for constructing such a communication network.

Answer:

Here minimum spanning algorithm is required to be used. where ever space station is node of undirected graph, every edge is denoted by distance between stations.:

Algorithm KruskalMST(G):

Here the input is a simple connected weighted graph G with n vertices and m edges and the output is minimum spanning tree T for G

for each vertex v in G do
Define an elementary cluster $C(v) \leftarrow \{v\}$.
Let Q be a priority queue storing the edges in G, using edge weights as keys
$T \leftarrow \emptyset$ // T will ultimately contain the edges of the MST
while T has fewer than $n - 1$ edges do
$(u, v) \leftarrow Q$.removeMin()
Let $C(v)$ be the cluster containing v
Let $C(u)$ be the cluster containing u
if $C(v) = C(u)$ then
Add edge (v, u) to T
Merge $C(v)$ and $C(u)$ into one cluster, that is, union $C(v)$ and $C(u)$
return tree T

2. (12 Points) Suppose you are given a network, G, of routers (vertices) and links (edges), with a source, s, and sink, t, together with bandwidth constraints on each edge, which indicate the maximum speed MB/s that the communication link represented by that edge can support. As in a Maximum Flow Algorithm, your goal is to produce a maximum flow from s to t, respecting the bandwidth constraints on the edges. Suppose now, however, that you also have a bandwidth constraint on each router in the network, which specifies the maximum amount of information MB/s that can pass through that router. Describe an efficient algorithm for finding a maximum flow in the network, G, that satisfies the bandwidth capacity constraints on the edges as well as the vertices. What is the running time of your algorithm?

Answer:

The Ford- Fulkerson Algorithm can be used for maxflow problem
Ford-Fulkerson Algorithm follows a simple idea-
It starts with initial flow as 0. While there is an augmenting path from source to the sink the flow to flow path is added. And the flow is returned.

```
Flow = 0
For each edge (u, v) in G:
        Flow (u, v) = 0
While there is a path p from s → t in residual network G_f:
        residual_capacity(p) – min(residual_capacity(u,v): for (u, v) in p)
        flow = flow + residual_capacity(p)
        for each edge (u, v) in p:
                if (u, v) is a forward edge:
                        Flow (u, v) = flow (u, v) + residual_capacity (p)
                else
                        Flow (u, v) = flow (u, v) - residual_capacity (p)
        return flow
```

here Augmentation can be done in O€ and the total worst case running time O(E|f*\) where f* is the max-flow found by the algorithm. If all flows are integers, then the while loop of For-Fulkerson is run at most times. This is because the flow is increase, at wordt by 1 in each iteration.

3. (12 Points) We define SUBGRAPH-ISOMPRPHISM as the problem that takes a graph G, and another graph H, and determine if H is isomorphic to a subgraph of G. That is the problem is to determine whether there is one-to-one mapping, f, of the vertices in H to a subset of the vertices in G such that if (u, w) is an edge in H, then (f(u), f(w)) is an edge in G. Show that SUBGRAPH-ISOMPRHISM is NP-complete.

Answer:

Let (G_1, G_2) be an instance and assume the nodes are numbered 1, 2, 3, 4, …. |G|
Let X be a witness, and list of nodes i_1, i_2, i_3, i_4, … in where n = $|G_1|$

The next step is to verify G_1 is isomorphic to a subgraph G_2 which means verifying that the subgraph induced by the node list x is equal to G_1.
This step can be executed by for each i_k, i_l verify that (i_k, i_l) belongs to E_1 if and only if (k,l) belongs to E_2.

Here as there are O (n^2) such pairs, the check would take the polynomial time, which would prove that subgraph isomorphism is in NP.

The next step is to show that the Subgraph-Isomorphism is NP_ Hard. This can be done by using the Subgraph-Isomorphism to solve the known NP-complete problem CLIQUE. Here, let (G, K) be an instance of clique where we want to determine whether the graph G contains a clique size k. The instance of Subgraph-Isomorphism (G, Gk) is created | Gk is clique with k nodes. Now it is shown that these two instances are equal.

If G has a k-clique, then G has the complete k-graph as subgraph along with the instance (G, Gk) which is a yes instance. If G had Gk as subgraph, then it has a k-clique as subgraph, and hence (G,k) is a yes- instance of CLIQUE.
It can be concluded that Subgraph-Isomorphism is NP-Hard and hence NP-complete.

4. (12 Points) Suppose you are processing a large number of operations in a consumer-producer process, such as a buffer for a large media stream. Describe an external-memory data structure to implement a queue so that the total number of disk transfers needed to process a sequence of n ***enqueue*** and ***dequeue*** operations is O(n/B).

Answer:

Here external-memory inefficient dictionary implementations based on sequences are considered. If the sequences representing a dictionary is implemented as an unsorted, doubly linked list, then insert & remove can be performed with O (1) transfers each, such that each block holds an item to be removed.
This searching process requires O (n) transfers in worst case, since each link hop that is performed could access a different block. The search time can be improved to O (n/B) transfers where n denotes the n Enqueue and Dequeue operations, and B denotes the number of nodes of the list that can fit into a block.

Alternatively, here the sequence using sorted array can be implemented. In this case, a search performs O (log2 n) transfers using the binary search but, this requires O (n/B) transfers to implement an insert or remove operation in the worst case as all the blocks have to be accessed to move the elements up or down.

Hence, sequence dictionary implementations are not efficient for external memory.

The dictionary queries and updates can be performed using O (log B n) = (O log n / log B) transfers. The main idea for improving the external-memory performance of the dictionary implementation is to perform up to O (B) internal-memory accesses to avoid a single disk transfer, where B denotes the size of a block. The disk performs this many internal-memory accesses just to bring a block into internal memory, and this is only a small fraction of the cost of a disk transfer.

Thus, O (B) high-speed, internal-memory accesses are a small price to pay to avoid a time-consuming disk transfer.
To reduce the importance of the performance difference between internal-memory accesses and external-memory accesses for searching dictionary can be represented using a multi-way search tree, which is generalization of the (2,4) tree data structure to a structure known as an (a, b) tree. Hence Buffered repository tree is a structure whose insertion cost is lower than its lookup cost.

5. (12 Points) Design an $O(n^2)$ time algorithm for testing whether a polygon with n vertices is simple. Assume that the polygon is given by the list of vertices.

Answer:

Algorithm Polygon(V):

Here the input is list of vertices V with all vertices of polygon and output is whether the polygon is simple or not.

VV is the list of pair of end points of line So,

For p1 in V,

For p2 in V,

If (p1, p2) or (p2, p1) in VV,

Continue

Else

If p1 is adjacent to p2,

VV.insert((p1,p2))

// p1 and p2 are the points in a line.

For (p1, p2) in VV,

// q1 and q2 are the points in a line.

For (q1, q2) in VV,

If ((p1, q1, p2) = Counter_Clockwise_CCW and (p1, q1, q2) = Clockwise_CW and (p2, q2, p1) = Clockwise_CW and (p2, q2, q1) = Counter_Clockwise_CCW or ((p1, q1, p2) = Collinear_COLL and (p1, q1, q2) = Clockwise_CW and (p2, q2, p1) = Clockwise_CW and (p2, q2, q1) = Counter_Clockwise_CCW)) then

Return false

Return true

6. (12 Points) Show how to modify the KMP string pattern matching algorithm so as to find every occurrence of a pattern string P that appears as a substring in T, while still running in O(n + m) time. (Be sure to catch even those matches that over- lap.)

Answer:

Here, instead of returning when a match is found, the index is stored &
i = i +1 and j = f (m-1)
The modified KMP algorithm would be:

```
Algorithm KMPMatch (T, P)
        Failure = failureFunction(P)l
        i = 0, j=0;
//Here an empty stack is constructed which stores the location of all the
occurrences of P in T.
Construct an empty stack S;
while (i < n)
        if (T[i] = P[j]) {
                if (j = m-1) {
                        S.push(i-j);
//An occurrence and (i-j) is the location of the given occurrence
i = i+1;
j = f(m-1);
}
else {
        i = i + 1;
        j = j + 1;
    }
else
        if (j > 0)
                j =F[j-1];
        else
                i = i + 1;
        return S; // if S is empty then no match is found
}
```

7. (12 Points) Given a variation of Algorithm 19.2 (randomSort) on Page 532 that runs in O(n) time with probability of $1-1/n^4$

Answer:

Algorithm randomSort(*X*):
Input: A list, *X*, of *n* elements
Output: A permutation of *X* so that all permutations are equally likely
Let *K* be the smallest power of 2 greater than or equal to *n*3
for each element, *xi*, in *X* do
Choose a random value, *ri*, in the range [0, $K - 1$] and associate it with *xi*
Sort *X* using the *ri* values as keys via radix-sort
if all the *ri* values are distinct then
return *X* according to this sorted order
else
Call randomSort(*X*)
Algorithm 19.2: A sorting-based algorithm for generating a random permutation.

8. (8 Points) Suppose a set S contains n two-dimensional points whose coordinates are all integers in the range [0, N]. What is the worst-case depth of a quadtree defined on S? Justify your answer.

Answer:

The worst-case depth of quadtree defined on S will of N. This is the case because the quadtree as defined by book is created with recursively performing a split at each child. They should be one in region. Therefore, the worst case depth is log4 = log4N2 = logN. (N is the range of one side)

9. (8 Points and 3-point Bonus) A chocolate factory has two products, white and dark chocolate. How much of each should it produce to maximize the profit? Assume it makes x_1 boxes of white and x_2 boxes of dark per day, with a profit of $1 on a white box, and a profit of $6 on the dark box.
The daily demand for these chocolates is limited to at most 200 boxes of white and 300 boxes of dark.
Finally, the current work force can produce at most 400 boxes of chocolate per day.
(a) Your job is to present this story as a Linear Program in standard form.
(b) Then present it in slack form.
You are not required to solve the problem, but if you provide the correct answer you get 3 bonus point.

Answer:

Standard form: Maximize the values first
$Z = X_1 + 6X_2$ which is subject to following: -
$X_1 \leq 200$, $X_2 \leq 300$, $X_1 + X_2$;
$X_1, X_2 \geq 0$

Slack Form Maximize the values first
$Z = x_1 + 6X_2$ which is subject to following: -
$X_3 = 200 - X_1$, $X_4 = 200 - X_2$, $X_5 = 200 - X_1 - X_2$;
$X_1, X_2, X_3, X_4, X_5 \geq 0$
There are free variables: FV = {1, 2}
Basic variables BV = {3, 4, 5}
Basic Solution: (0, 0, 400, 200, 300)

Now Simplex method is used to solve: The value of X_1 is raised and the constraints on X_1 are $X_1 \leq 400$, $X_1 \leq 200$ and $X_1 \leq \infty$. The tightest of these constraints is $X_1 \leq 200$ coing from the basic variable X_4. So $X_1 = 200$ is set and $X_4 = 0$ is set by pivoting X_1 and X_4 which would yield and new slack. Where the values are first maximized $Z = 200 - X_4 + 6X_2$ which is subject to $X_3 = 200 + X_4 - X_2$, $X_1 = 200 - X_4$, $X_5 = 300 - X_2$;
$X_1, X_2, X_3, X_4, X_5 \geq 0$
There are free variables: FV = {2, 4}
Basic variables BV = {1, 3, 5}
Basic Solution: (200, 0, 200, 0, 300)
Objective Value: C*= 200

Here the coefficient on X_4 is negative, raising that value from zero will only decrease the objective function. So X_4 is fixed at 0 and raise X_2 until the first constraint is hit. The constraints on X_1 are: $X_2 \leq 200$, $X_2 \leq 300$, $X_2 \leq \infty$. The tightest of these constraints is $X_2 \leq 200$ coming from the basic variable X_3 and hence $X_2 = 200$ and $X_3 = 0$ is set by pivoting X_2 and X_3. Which yields another slack. Where the values are first maximized $Z = 1400 + 5X_4 + 6X_2$ which is subject to $X_2 = 200 + X_4 – X_3$, $X_1 = 200 - X_4$, $X_5 = 100 – X_4 + X_3$;
$X_1, X_2, X_3, X_4, X_5 \geq 0$
There are free variables: FV = {3, 4}
Basic variables BV = {1, 2, 5}
Basic Solution: (200, 200, 0, 0, 100)
Objective Value: C*= 1400

Here the coefficient on X_3 is negative, raising that value from zero will only decrease the objective function. So X_3 is fixed at 0 and raise X_4 until the first constraint is hit. The constraints on X_4 are: $X_4 \leq 200$, $X_4 \leq 100$, $X_4 \leq \infty$. The tightest of these constraints is $X_4 \leq 100$ coming from the basic variable X_5 and hence $X_4 = 100$ and $X_5 = 0$ is set by pivoting X_4 and X_5. Which yields another slack. Where the values are first maximized $Z = 1900 + 5X_5 + X_3$ which is subject to $X_2 = 300 – X_5$, $X_1 = 100 - X_5 – X_3$, $X_4 = 100 – X_5 + X_3$;
$X_1, X_2, X_3, X_4, X_5 \geq 0$
There are free variables: FV = {3, 5}
Basic variables BV = {1, 2, 4}
Basic Solution: (100, 300, 0, 100, 0)
Objective Value: C*= 1900