95

Name(last, first): Rubenacker, Sam

UCLA Computer Science Department

iA

CS 180 **Algorithms & Complexity** ID : 504295581

Midterm **Total Time: 1.5 hours** **October 27, 2016**

Each problem has 20 points.

All algorithm should be described in English, bullet-by-bullet

1 a. Describe Depth First Search on an undirected and un-connected graph.
b. Analyze the time complexity of DFS when there are no cycles.

a)

- choose an arbitrary starting node v
  - push all of v's neighbors onto a stack
  - ~~for each node w in the stack~~
    - ~~recursively run DFS on w~~

+15

  - pop nodes "w" off the stack until the stack is empty
    - recursively run DFS on the new subtree starting at w, if w has not already been visited
- if all nodes have been visited at this point, we are finished
  - if not, arbitrarily choose another node that has not been visited, and run DFS on this node

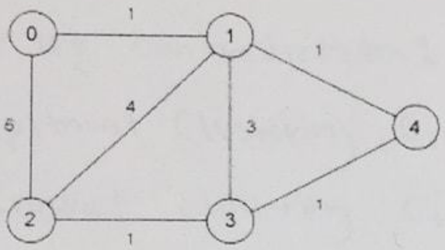
b) The time complexity is $O(n+e)$, where n is the number of nodes, and e is the number of edges. Each edge is visited once, which is $O(e)$. In the case of an unconnected graph, we must also visit nodes not connected by edges, so we must also visit every node once. Therefore, total time complexity is $O(n+e)$.

+5

20

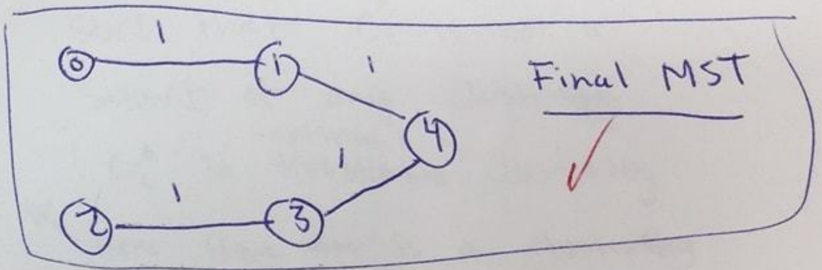
Name(last, first): Rubenacker, Sam

2 . a. Use Prim's MST algorithm to find an MST in this graph. Show each step on the graph shown below.
b. If some edges were negative would the algorithm still find an MST. Why?



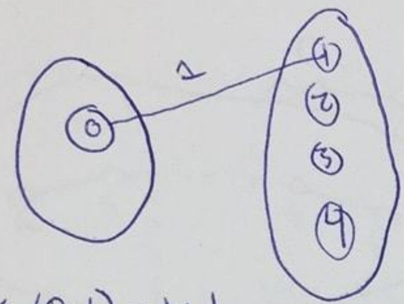
+15

For each partition, e_{min} is the smallest edge connecting the two partitions, and will be in the MST, by the MST theorem.



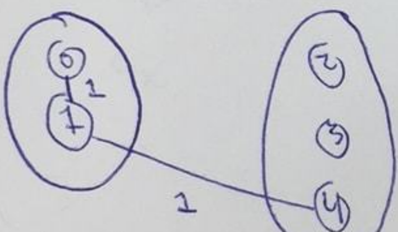
a)

1st partition



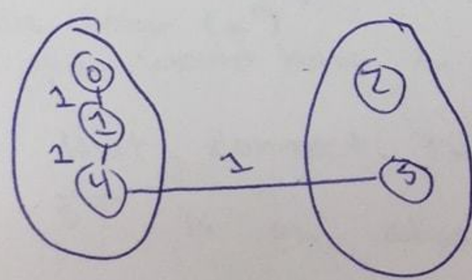
e_{min} is $(0,1) = |1|$

2nd partition



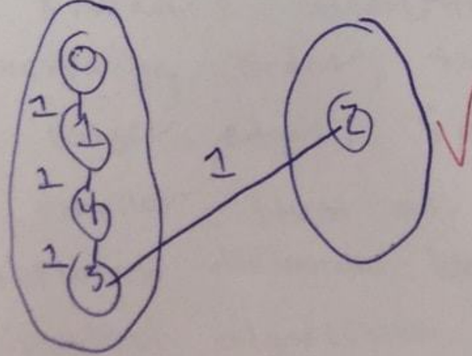
e_{min} is $(1,4) = |1|$

3rd partition



e_{min} is $(4,3) = |1|$

4th partition



e_{min} is $(3,2) = |1|$

b) If some edges were negative, the algorithm would still find an MST. This is because we are concerned with finding the smallest edge between 2 partitions, and this smallest edge e_{min} is guaranteed to be in the MST. Negative edges become a problem in other greedy algorithms concerning finding shortest paths, such as Djikstra's.

+5

20

Name(last, first): Rubenacker, Sam

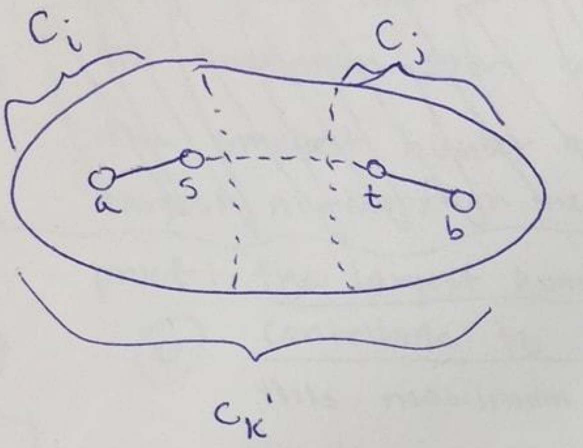
20

3. Prove that solving the k-clustering problem (described in class and in the book) using Kruskal's MST algorithm, produces an optimal clustering. That is, it will produce an optimal set of clusters C1, C2, …, Ck with Maximum cluster distances. (Use a figure to better describe your proof: as was done in class / book).

Proof by contradiction:

Optimal Clustering $C_1, C_2, \dots, C_K$ (C)

Kruskal clustering $C_1', C_2', \dots, C_k'$ (C') (non optimal)



There is a ~~clustering~~ C_i' such that C_i' is not a subset of any ~~clustering~~ C_i in optimal ~~Kruskal's~~ clustering.

Here there ~~are~~ is a ~~clustering~~ C_K' which is not a subset of any subsets C_i, C_j in the optimal clustering.

By Kruskal's algorithm, edge $\overline{ST}$ is an internal edge (of the cluster C_K'); and would have a smaller weight than any edges that connect two neighbor clusters. Here, edge $\overline{ST}$ is an edge connecting two clusters C_i and C_j in the "optimal clustering". This cannot be the case, since Kruskal's algorithm adds edges based on their weight in increasing order, such that smaller edges are added before larger edges. Since $\overline{ST}$ is an internal edge, it is smaller than an edge which would be the max of the minimum distance between clusters. Therefore, the ~~optimal~~ ~~clus~~ Kruskal algorithm is optimal.

15

Name(last, first): Rubenacker, Sam

4. Consider a sequence of n real numbers X = (x1, x2, …, xn).
a. Design an algortihm to partition the numbers into n/2 pairs. We call the sum of each pair S1, S2, …, Sn/2. The algorithm should find the partitioning that minimizes the maximum sum.
b. Analyze the time comlpexity of your algorithm.

1 2 3 4 5 6

7 7 11

7

1 3 5 9 17 21

2 4 6 8 100 1000

a) algorithm:

- first sort the sequence X in non decreasing order
- ~~the pair that will minimize the maximum sum will be the smallest number and the largest number from the sorted sequence~~

continued ON BACK

proof: the largest number in the sequence will always contribute to the maximum sum. To minimize this maximum sum, pair the largest number in the sequence with the smallest number, so that you can minimize the sum.

0

10

Example 1, 5, 6, 7

b) time complexity:

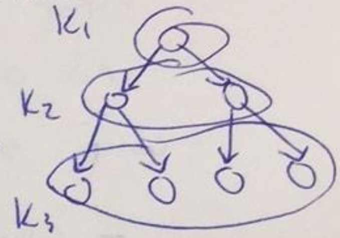
5

- it takes O(n log n) to run mergesort on the sequence X
- it then takes O(n) to go through the list ~~finding the maximum pair~~ creating the pairs
- ~~so total time is O(n log n) dominated by the sort~~
- another (n-1) to find the max
- total time is O(n log n), dominated by the sort

2

Name(last, first): Rubenacker, Sam

5. Let G be a DAG and let K be the maximal number of edges in a path in G.
a. Design an algorithm that divides the vertices into at most k+1 groups such that for each two vertices v and w in the same group there is no path from v to w and there is no path from w to v.
b. Analyze the time complexity of your algorithm.



a) bipartite, K-coloring, ~~BFS layers form groups~~

~~algorithm:~~

~~- run a directed BFS on G~~
~~- each layer i explored by the BFS will correspond to a group K~~

~~proof:~~
~~- since it is a DAG, there are no cycles in the graph~~

algorithm:

- run a topological sort on G
- every time you delete sources from G, add these source nodes S_i to a group K_i
- when you have no more nodes in the graph, you will be left with at most K+1 groups, which were formed by deleting sources

proof:

- since it is a DAG there are no cycles
- for any group K_i, made of sources deleted on the same iteration of topological sort, there are no paths between any nodes v and w in the group
- a source was deleted because it has in-degree of 0, so there is no way there could be a path to this source from another source

ON BACK