CS2010 Semester 1 2012/2013

Data Structures and Algorithms II

Tutorial 05 - Graphs 2

For Week 07 (01 October - 05 October 2012)

Released: Wednesday, September 26, 2012

Document is last modified on: October 3, 2012

1 Introduction and Objective

Welcome back from recess week =). I hope you are a bit fresher now. Note: I know that if you had aimed for higher marks by attempting PS bonus, your recess week was probably not a recess week at all...

Week07 is usually a mid-semester test week in NUS. However, CS2010 is different. We had run our Quiz1 two weeks ago (Week06) and therefore this week, we have a (supposedly) lighter workload.

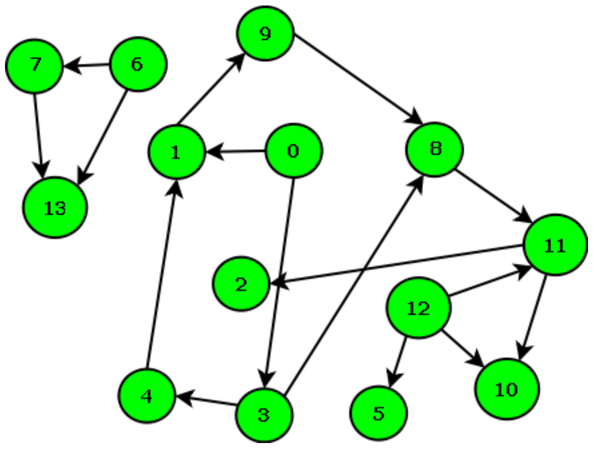
In this tutorial, we will resume our discussion on Graph problems. We have one question from Lecture 5 (Topological Sort) and the rest are from Lecture 6 (Minimum Spanning Tree). We will also discuss PS4 Subtask 1 during this tutorial.

Note: Use `http://www.comp.nus.edu.sg/∼stevenha/visualization/mst.html` to *verify* the answers of some questions in this tutorial. However during written tests, you have to be able to do this by yourself.

2 Tutorial 05 Questions

Topological Sort

Q1. Give the topological ordering of vertices of the DAG shown below. Although there are other algorithms to do topological sort, please use the DFS based algorithm as shown in Lecture 05. You can assume that if DFS encounters a situation where it has to choose one of several valid vertices, prefer the lower vertex number first. We should have a unique answer.



$$\begin{tikzcd}
6 \arrow[r] \arrow[d] & 7 \arrow[d] \\
13 & 13
\end{tikzcd}$$

Figure 1:

Ans:
The sequence of nodes traversed and the (reversed) output produced by DFS:

0,1,9,8,11,2 $\rightarrow$ 2
0,1,9,8,11,10 $\rightarrow$ 10
0,1,9,8,11 $\rightarrow$ 11
0,1,9,8 $\rightarrow$ 8
0,1,9 $\rightarrow$ 9
0,1 $\rightarrow$ 1
0,3,4 $\rightarrow$ 4
0,3 $\rightarrow$ 3
0 $\rightarrow$ 0

5 $\rightarrow$ 5

6,7,13 $\rightarrow$ 13
6,7 $\rightarrow$ 7
6 $\rightarrow$ 6

12 $\rightarrow$ 12

The topological ordering is thus $= \underline{12,}\ \underline{6,\ 7,\ 13,}\ \underline{5,}\ \underline{0,\ 3,\ 4,\ 1,\ 9,\ 8,\ 11,\ 10,\ 2}$

Minimum Spanning Trees

Q2. In Lecture 06, you are presented with two MST algorithms: Prim's and Kruskal's algorithms. List down the similarities and differences of these two algorithms!

Ans: Similarities: Both are graph algorithms that can solve the MST problem, Both runs in $O(E \log V)$. Both are greedy algorithms. (Other similarities not listed here are possible).
Differences: The inventors are different, and basically: the way each algorithm works (see below):

In Kruskal's, adding edges one by one can cause **multiple components/tree** to spring up as the algorithm progresses at any point in time (thus making a forest), when we add a new edge we are not sure whether that edge links 2 different trees together or they cause a cycle in an existing component. Thus cycle detection is not as easy as just checking whether the 2 nodes spanned by the edge have already been include in the MST (we need another DS: the Union-Find Disjoint Sets to do this efficiently). In the end, all the different components will be linked up to form one single component.

In Prim's algorithm, the final MST is built by growing it one edge at a time from a **single tree**. Thus at any point in time, cycle detection is easy. We can just check whether the nodes spanned by an edge are already included in the tree. If they are, then including the edge will form a cycle and we can ignore it. If not, we can safely include that edge.

Q3. Give the MST of the following undirected graph, using both Prim's and Kruskal's algorithm. In Prim's algorithm, please start from node A and break ties by preferring vertices with smaller vertex label. In Kruskal's algorithm, break ties by preferring edges with smaller vertex pair labels (e.g. (A, F) is smaller than (B, C)). Show how the MST is built step by step in each case.

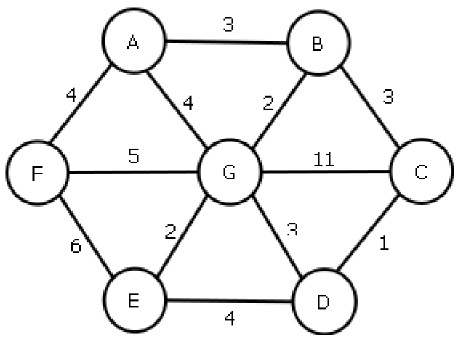


Figure 2:

Ans:

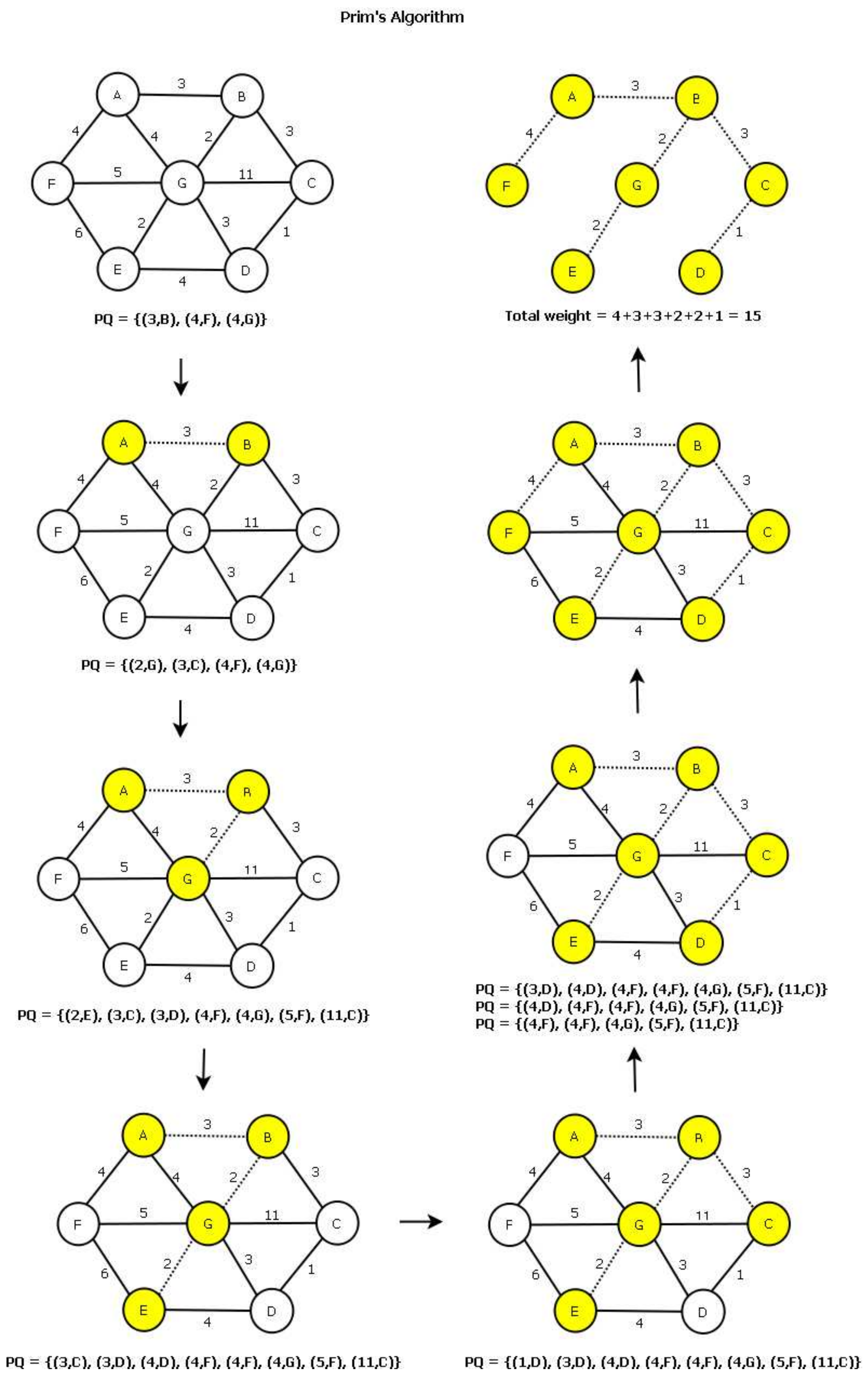

Prim's Algorithm
PQ = {(3,B), (4,F), (4,G)}
PQ = {(2,G), (3,C), (4,F), (4,G)}
PQ = {(2,E), (3,C), (3,D), (4,F), (4,G), (5,F), (11,C)}
PQ = {(3,C), (3,D), (4,D), (4,F), (4,F), (4,G), (5,F), (11,C)}
PQ = {(1,D), (3,D), (4,D), (4,F), (4,F), (4,G), (5,F), (11,C)}
PQ = {(3,D), (4,D), (4,F), (4,F), (4,G), (5,F), (11,C)}
PQ = {(4,D), (4,F), (4,F), (4,G), (5,F), (11,C)}
PQ = {(4,F), (4,F), (4,G), (5,F), (11,C)}
Total weight = 4+3+3+2+2+1 = 15

Figure 3:

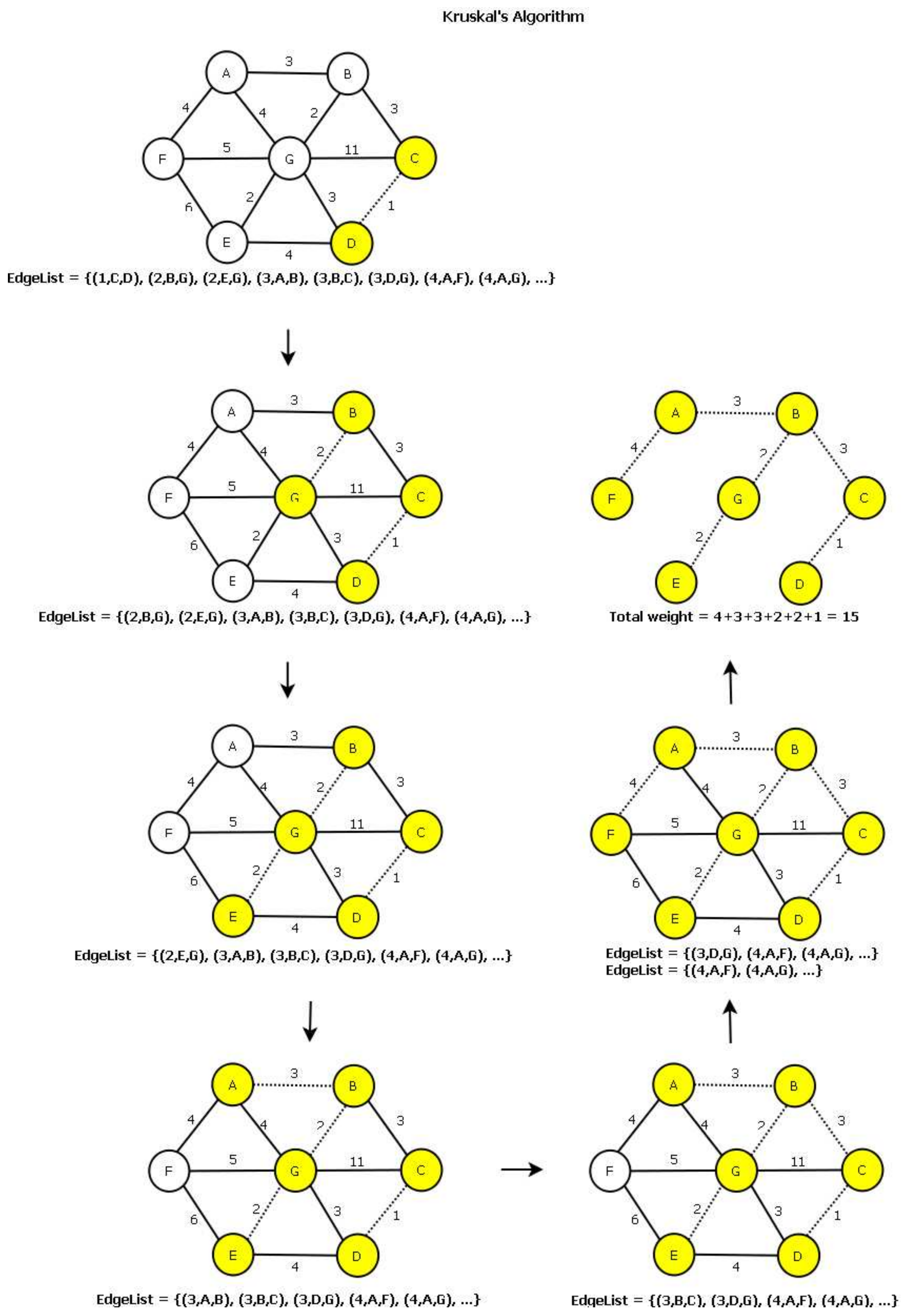

Kruskal's Algorithm
EdgeList = {(1,C,D), (2,B,G), (2,E,G), (3,A,B), (3,B,C), (3,D,G), (4,A,F), (4,A,G), ...}
EdgeList = {(2,B,G), (2,E,G), (3,A,B), (3,B,C), (3,D,G), (4,A,F), (4,A,G), ...}
EdgeList = {(2,E,G), (3,A,B), (3,B,C), (3,D,G), (4,A,F), (4,A,G), ...}
EdgeList = {(3,A,B), (3,B,C), (3,D,G), (4,A,F), (4,A,G), ...}
EdgeList = {(3,B,C), (3,D,G), (4,A,F), (4,A,G), ...}
EdgeList = {(3,D,G), (4,A,F), (4,A,G), ...}
EdgeList = {(4,A,F), (4,A,G), ...}
Total weight = 4+3+3+2+2+1 = 15

Figure 4:

Q4. An ambitious cable company has obtained a contract to wire up the government offices in the city with high speed fiber optics to create a high speed intra-net linking up all the different governmental departments. In the beginning they were confident that the minimum cost of connecting all the offices will be within budget. However, they later found they made a miscalculation, and the minimum cost is in fact too costly. In desperation, they decided to group the government offices into K groups and link up the offices in each group, but not offices between groups to save on the cost. This effectively creates K intra-nets instead of one big intra-net.

Given a budget B, V government offices, and the cost of linking up any given pair of offices, help the company design a program which will tell them what is the smallest value of K (so as to minimize the number of intra-nets), what are the offices in each of the K group and how they should be linked such that the total cost of all linking is minimized in each group and also within budget.

The program should model the problem as a graph. It should also run in $O(E \log V)$ time. Where E and V are the number of edges and vertices in the graph, respectively.

*Can you prove why your program gives the smallest number of groups such that the total cost of linking up offices is minimized and also within budget?

Ans:

Algorithm :-

1. Perform Kruskal's, keep track of sum of the weights of the edges picked so far. Once an edge that is going to be added will cause the sum to exceed the budget B, stop the algorithm. This is $O(E \log E)$ or $O(E \log V)$. Then report K = the number of disjoint sets currently in the UFDS (or use step 2 and 3 below).
2. Remove all edges not picked by Kruskal's from the graph. This is $O(E)$.
3. Find the number of components by running BST until all vertices have been processed. This is $O(V + E)$.

Overall = $O(E \log V)$

*Proof:

Proof that linking the K groups is within budget :-
The MST $T*$ built using Kruskal's has the minimum cost $C(T*)$. If this cost is still over budget, the algorithm does the equivalent of picking the largest edge to remove, thus forming 2 spanning tree $T1$ and $T2$. It will keep removing edges and creating more spanning trees until the sum of the cost of the remaining edges is within budget.

Proof that K is the minimum number of groups created :-
(Number of edges removed)+1 = (number of groups created). Let $K*$ be the minimum number of groups created and $C(K*)$ be the sum of the cost of linking within each group.
$C(T*) - C(K*)$ is the difference in cost that should be covered by the removed edges.
Our aim is to create a set of deleted edges D such that $C(D) \geq C(T*) - C(K*)$ and $|D|$ is minimized. Since the algorithm removes edges based on decreasing order of cost until the budget is met, the set of edges picked are the $K - 1$ largest edges in $T*$.

Assuming this set is not the smallest set, that means we can either

1. remove some edges from the set and still meet the budget. or

2. remove n edges from the set and replace it by m edges where $m < n$.

For 1., according to the algorithm this is not possible since the $K - 1$ edges we remove are the 1st largest $K - 1$ edges which meets the budget.

For 2., $C(m) \leq C(n)$ for
any m edges from the unpicked edges, since the m edges must come from the (K)th cost edge onwards.

Proof that the linkage for each group is the MST for that group :-
Assuming edge e is the 1st edge picked to be removed such that the spanning tree of one or both of the resulting 2 groups created is not the MST for that group. Now that means there should be another edge $e*$ which should have been removed instead where $C(e*) > C(e)$. However at any point, the edge e picked to be removed is the largest remaining edge, thus this cannot happen.

Q5. Please download CS2010 Final Exam paper S1, AY2011-2012 (last year's final exam) and solve a problem titled: **Vehicle Monitoring System (20 marks)**

Ans: Simply run Kruskal's but sort edges in non-increasing (downwards) order. You will get Max ST. Edges not taken by Kruskal's will be the edges where we have to place the cameras. Sum their weights! The time complexity is simply $O(E \log V)$, the same as Kruskal's time complexity.

The solution is very short but not easy to arrive at such solution. Your tutor will go through the 'thinking process' on how to arrive at such solution during the actual tutorial.

Problem Set 4

Q6. Discussion of PS4 Subtask 1

Ans: There is only one unique path in a tree. So Grace has no choice other than to traverse this unique path. The hardest corridor for Grace is simply the edge with the highest weight along this unique path. For each query, we can simply run a DFS (or BFS) from the starting point and traverse the graph, remembering the maximum edge weight encountered so far. We stop as soon as our DFS (or BFS) hits the target point. Time complexity is Q calls of DFS/BFS on a tree, which is $O(QV)$. This subtask has small Q and V so this should not be a problem.

Now, do you realize that if the given graph is a tree, this problem is easy?

If the given graph is not a tree (a general graph Subtask 2-5), can we use a certain tree of that graph to make the original problem simpler? Hint: What is the main theme of Lecture 06 (or this tutorial)?