Daniel Campos

November 1st, 2012

Algorithms section 5

HW 5

1. 4.13

Input: An underreacted graph G=(V,E), The cars gas capacity in amount of miles coverable so L.

Output: If it is possible to make it from city S to city T

Q=[s]//a queue containing just the start city

While Q is not empty:

U=eject (Q)//removes the first item in the queue from the edge

For all edges (E,V) belonging to the current Edge

If(l(e)<=L)// getting to the edge is possible by the gas

{

If(e == t)// city t has been reached

{ Return true;}

Else

{

Inject(Q,V);//city is not city t but is still reachable // so its added the queue and will be explored further

}

}

}

}

Return false// queue is empty and never returned true so the city was unreachable

Justification: This algorithm works since it uses the principle of the breadth first search and the idea of a queue but unlike the normal BFS it only adds the edge if it is reachable by the gas. This will also allow if, for example city D is reachable from E but not from C, It will try to add it from C and not be able to and then when the loop gets to edge E it will add it and end the programs since the city can be reached. Because the BFS algorithm is in linear time/O(|V|+|E|) and this algorithm is a slightly shorter version of it this algorithm will also be in linear time.

Alternatly you can also Remove all edges from the graph with length greater than L and then do a DFS starting at s and see if t can be reached. Both solutions work

2. 5.5

(a) Can the minimum spanning tree change?

No, it cannot. The weight of each spanning tree is increased by exactly |V| - 1. The identity of the lightest spanning tree is therefore preserved.

(b) Can shortest paths change?

Yes. Here's an example: when the edge lengths are increased, the shortest path from S to T changes from S ->A -> B ->T to S -> T.

A - - -5- - -B

| |

1 3

| |

S- - - 10- - -T

3. 5.7

procedure max\_span\_tree

Input: A connected undirected graph G=(V,E) with edge weight w(e)

Output: A max spanning tree defined by the edges X

for all e in V:

makeset(u)

x={}

make each weight of the each edge to another be 1/w(e) instead of w(e)

sort the edges E by weight// since the ones that had larger weights are now the smallest it will have the largest original weights re-written as the smallest

if (find(u)!= find(v)// not already in tree

add edge {u,v} to X

union(u,v)

//if it were needed to write the weight of each edge then the next step would be just to loop through every edge and multiply its weight by its reciprocal twice(e.g. (1/x)\*x\*x=x) and the original weights would now be correct

Justification: this is based on the kruskal algorithm but since when 1 is divided by a number the larger the division the smaller the result will be, those with the highest weight will end up with the smallest number(Example if original weights from point A to point B were 4,7,9,1 originally these would have been ordered 1,4,7,9. After the operation they would be ordered 1/9,1/7,1/4,1 and the original largest connection would now be the smallest.

4. 5.14

a) based on the division procedure the encoding of the letters would be as follows

a=0

b=10

c=110

d=1110

e=1111

b)Total amount of characters 1,000,000

Amount of A characters is =500,000. Since each a requires one bit per character then 500,000 bits are required for A

Amount of B characters is =250,000. Since each a requires two bits per character then 500,000 bits are required for B

Amount of C characters is =125,000. Since each a requires Three Bits per character then 375,000 bits are required for C

Amount of D characters is =62,500. Since each a requires four Bits per character then 250,000 bits are required for D

Amount of E characters is =62,500. Since each a requires four Bits per character then 250,000 bits are required for E

Therefor the total amount of bits required is 1875000

5. 5.21

A feedback edge set is an undirected graph is a set of edges whose removal renders the graph acyclic Give an efficient algorithm to find a feedback edge set of a minimum weight.

Assuming the graph is connected(otherwise run each component separately)

Given as an input graph G=(V,E) and edge weight w(e), find a maximum spanning tree of G by running kruskals algorithm on (G, {-w(e)});

Return all edges not in this spanning tree

Justification: by its definition a feedback edge set is any set of edges whose removal from G leaves behind a forest. The lightest feedback edge set therefore corresponds to the heaviest possible remaining forst namely the maximum spanning tree

Run time: run time of kruskals algorithm, O(|E|log|V|)