March 23, 2012

# 1 Solution of 3

## 1.1 Algorithm

Here is the complete algorithm for the problem:

$E' \leftarrow \Phi$
**while** there is any vertex left in the graph **do**
    Pick any vertex $v$ arbitrarily;
    Grow a BFS tree starting from $v$, level by level, and stop as soon as it reaches level $i$ such that the number of vertices at level $i$ is less than $n^{1/k}$ times the number of vertices at the level $i$ - 1;
    Let the BFS tree be grown to level $i$ in this manner;
    Add to $E'$ all those edges of the BFS tree upto level $i$;
    Remove all vertices of the BFS tree upto level $i$ - 1 from the graph;
**return** ($V$, $E'$)

## 1.2 Proof that the algorithm computes 2$k$ - 1 spanner

First, no BFS can be more than $k$ levels deep. This can be seen easily, at the first level we start with one vertex and we proceeed only if at the next level there are atleast $n^{1/k}$ vertices. So, the number of vertices at k$^{th}$ level is atleast:

$$\text{V} = 1 \text{ x } n^{1/k} \text{ x } n^{1/k} \text{ ... x } n^{1/k} = n^{k/k} = \text{n vertices}$$

which can not be true, so the BFS terminates before k levels. In a BFS tree all the non tree edges are between successive levels. When we discard all the vertices upto level $i$ - 1, we take all tree edges in $E'$ and discard all the non tree edges. The edges we are discarding are such that one of their end points lie in the BFS tree upto level $k$ - 1 and the other atmost at level $k$. Such that the maximum distance between these edges can atmost be 2$k$ - 1. All the other non trees edges which have been discarded the new distance between them is also less than 2$k$ - 1. After this step the algorithm is called recursively over the set of vertices and edges remaining. So, every time an edge is discarded, there is a path of length not more than 2$k$ - 1 between its vertices. Hence, the algorithm correctly computes a 2$k$ - 1 spanner.

## 1.3 Proof of size of the spanner being $O(n^{1+1/k})$

In a BFS traversal suppose we discard V nodes. Then the number of edges of last level we include can be atmost V$n^{1/k}$. In addition to those edges we also

include all the edges inside the BFS tree which are V - 1 in number. So the total number of edges over all the BFS's is of the order of:

$$\sum (V - 1 + n^{1/k}V) = \mathrm{O}((1 + n^{1/k})\sum V) = \mathrm{n}(1 + n^{1/k}) = \mathrm{O}(n^{1+1/k})$$

# 2 Solution of 4

## 2.1 Algorithm

visited($v$) ← True;
$u$ ← infinity;
**DFS′**($v'$)
    minedge ← weight($v'$, $v$);
    for each child $w$ of $v'$;
        if visited($w$) == False;
            DFS′($w$);
            if (weight($u$) < weight($w$, $v'$))
                minedge ← min(minedge, weight($w$, $v'$))
                add $u$ to $E'$;
            else
                minedge ← min(minedge, $u$);
                add ($w$, $v'$) to $E'$
    $u$ ← minedge;

## 2.2 Proof using induction

In this algorithm we only consider those edges for DFS which were present in the original MST and all the new edges. This holds because of the follwing argument. Suppose an edge which was not present in the previous MST is present in the new tree. Since it was not present in the original tree inclusion of it must have been creating a cycle. For every edge ($u$, $v$) in the original tree which is absent in new subtree a path between $u$, $v$ still exists. So, including that vertex will still create a cycle in new MST if it was creating a cycle in original MST. Hence any vertex which was absent in the original MST is also absent in new MST.

## 2.3 Statement

After the DFS′ at a vertex $w$ finishes minedge and u are the largest edges in the path from $v'$(parent of $w$) to $v$ and $w$ to $v$ respectively.

## 2.4 Base Case

Suppose we consider the induction steps in increasing order of DFN number. In the base case $w$ is a leaf, the for loop will do nothing and u gets assigned the edge from $w$ to $v$. The variable minedge will also store the edge from parent of $w$ to $v$. So both the properties are satisfied.

## 2.5 Induction Step

Suppose that in DFS$'(v')$, we are inside the for loop and $w'$ is the vertex which is considered last in DFS$'(w)$. By induction, minedge and $u$ are largest edges in the path from $w$ to $v$ and $w'$ to $v$ respectively. minedge is now assigned the larger of edges $u$ and $(w, v')$. After the for loop finishes, $u$ is assigned minedge. So after this, $u$ is the largest edge in the path from $w$ to $v$. Same lines of argument follow for minedge and minedge is the largest edge in path from $v'$ to $v$.

After this has been proved, in the if else statement inside the for loop, we always take the smaller edges and never the largest edge in the new MST. The largest edge amongst minedge , $u$ and $(w, v')$ is not taken, so the largest edge in the cycle is deleted. This uses the cycle property of MST. Since n - 1 new edges are discarded and each discarded edge belongs to a cycle whose all other edges are in the tree, therefore the new tree is also an MST.

## 2.6 Time Complexity

Since the DFS is done on a MST the DFS function will have to be called only n times. Also Hence the time complexity is O(n).