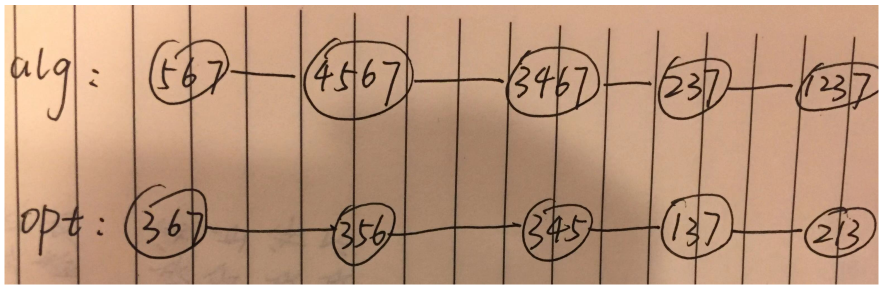
CS787: Advanced Algorithms **Scribe:** Group 4

Title: hw1

0.1 Tree Width

(A) Create new tree node N includes v and all v's neighbors. N = {v, neighbors of v} For all neighbors of v, say u_1, u_2, ...u_k, their corresponding subtrees in T' are T_1, T_2 ,..., T_k.
1. Pick an arbitrary tree node from T1, say N_1. Connect N to N_1.
2. If $T_2 \backslash T_1 \neq \emptyset$, then pich an arbitrary tree node from $T_2 \backslash T_1$, say N_2, and connect N to N_2. If $T_2 \backslash T_1 = \emptyset$, go to next step.
...
If $T_k \backslash (\bigcup_{j=1}^{k-1} T_j) \neq \emptyset$, then pick an arbitrary tree node from $T_k \backslash (\bigcup_{j=1}^{k-1} T_j)$, say N_k, and connect N to N_k. If $T_k \backslash (\bigcup_{j=1}^{k-1} T_j) = \emptyset$, go to next step.
We add a tree node of size $deg(v) + 1$ for every iteration in this way.

(B) the minimum possible tree-width for this graph is 2, because it is a Series-Parallel graph. Tree decomposition implemented by our algorithm and the optimal tree decomposition:



(C)

0.2 Traveling Salesman Problem

(A) Consider a very large complete graph with even vertex count n, the weight of each edge is 1.

$$cost(Algo) = cost(MST) + cost(Perfect\ Match) = (n-1) + (n/2) \simeq (3/2) * n = (3/2) * cost(OPT) \tag{0.2.1}$$

We find a instance for which Algo(I)/OPT(I) = α. Hence, the analysis is tight.

(B) Consider a complete graph with 3 vertex count.

$$LB = max(MST, 2 * MATCH) = max(2, 0) = 2 = (2/3) * 3 = (2/3) * cost(OPT) \tag{0.2.2}$$

0.3 Vertex Cover

(A) Consider a graph with vertice u in the middle, all other vertices are connect and only connect to u. The size of the graph is n.
The optimal solution is {u}. We may get {all vertices other than u} as a result by using this algorithm, and the size is n-1. And it's Ω(n) times larger than {n}.

(B) Consider any vertex v in the optimal vertex cover, and let N(v) be the set of the neighbors of v as well as v itself.

$$E[|S \cap N(v)|] = 1 * 1/2 + 2 * 1/4 + 3 * 1/8 + ... + n * (1/2)^n \tag{0.3.3}$$

When n $\to \infty$, $E[|S \cap N(u)|] \to 2$, you can prove this by multiply both sides by 1/2.
Thus $E[|S \cap N(v)|] < 2$, $E_(alg) < 2|OPT|$. Thus this variant achieves a 2-approximation.

(C) Consider graph with only two vertices u, v, and one edge (u,v). w$_u$ = 0 and w$_v$ = 1. The optimal solution is u. But the original algorithm will give answer u, v. The two variants may give answer v. Thus, all of them do not work for this problem.

(D) Consider any vertex v in the optimal vertex cover, and let N(v) be the set of the neighbors of v as well as v itself.

$$\begin{aligned} E[|S \cap N(v)|] = w(v) * (1 - p_1) + (w(u_1)) + w(v)) * p_1(1 - p_2) + ... \\ +(\sum_{1<=i<=n-1} w(u_i) + w(v)) * \prod_{1<=i<=n-1} p_i * (1 - p_n) \\ + \sum_{1<=i<=n} w(u_i) * \prod_{1<=i<=n} p_i \\ = w(v) * (1 - \prod_{1<=i<=n} p_i) + \\ w(u_1) * p_1 + w(u_2) * p_1 p_2 + ... + w(u_n) * \prod_{1<=i<=n} p_i \end{aligned} \tag{0.3.4}$$

Firstly, we have $w(v) * (1 - \prod_{1<=i<=n} p_i) < w(v)$

Secondly, since $p_i = w(v)/(w(u_i) + w(v))$ and $w(v) * \prod_{1<=i<=n} p_i > 0$,
then $w(u_n) * \prod_{1<=i<=n} p_i + w(v) * \prod_{1<=i<=n} p_i = w(v) * \prod_{1<=i<=n-1} p_i$, thus
$w(u_1) * p_1 + w(u_2) * p_1 p_2 + ... + w(u_n) * \prod_{1<=i<=n} p_i + w(v) * \prod_{1<=i<=n} p_i = w(v)$
thus we have $w(u_1) * p_1 + w(u_2) * p_1 p_2 + ... + w(u_n) * \prod_{1<=i<=n} p_i < w(v)$

So $E[|S \cap N(v)|] < 2 * w(v)$
Thus $E[\sum_{v \in S} w(v)] <= \sum_{v_i \in opt} E[|S \cap N(v_i)|] < 2W_(opt)$

0.4 Linear Programming

Consider a directed graph G(V,E), $l_{u,v}$ is the weight of edge (u, v). Denoting $X_{u,v}$ as the indicator variable, which ranges in [0, 1], meaning how much fraction on a (u, v) $\in$ E is chosen. The LP could be:

Objective function:

$$Minimize: \sum_{(u,v)\ in\ E} X_{u,v}\ *\ l_{u,v} \tag{0.4.5}$$

Constraints:

$$\sum_{(s,v)\ in\ E} X_{s,v} = 1 \tag{0.4.6}$$

$$\sum_{(u,t)\ in\ E} X_{u,t} = 1 \tag{0.4.7}$$

$$\sum_{for\ all\ predecessor\ x\ to\ y} X_{x,y} = \sum_{for\ all\ successor\ z\ to\ y} X_{y,z} \tag{0.4.8}$$

If there is unique shortest path, all $X_{u,v}$ must be integral, the set of all $X_{u,v}$ that equals to 1 indicates all (u, v) that construct this unique shortest path. Thus, the optimal cost by LP is the shortest path length.
If there are multiple shortest paths, and LP find integral solution on $X_{u,v}$, indicating one among all shortest paths. The optimal cost by LP is still the shortest path length.
If there are multiple shortest paths, and LP find fractional solution on $X_{u,v}$, indicating all shortest paths. But the optimal cost by LP is equal to one shortest path length.
To sum up the above three scenarios, LP finds the shortest path length.