CS 330: Homework 6

Michael Lin

December 3, 2015

Problem 1

(a) Suppose we had all points in P in a graph $G(V, E)$. Fix $\delta > 0$, and choose any two arbitrary points p_1, q_1. If there is an edge between p_1 and q_1, then we are done. If not, then by part (b) of Well-Separated Pair Decomposition, the 2 points form a pair (A_1, B_1) from the family Φ that is δ-separated. That means, there are points $a_1 \in A_1$ and $b_1 \in B_1$ such that $(a_1, b_1) \in E$.

Now, we argue that p_1, a_1 are connected, and q_1, b_1 are also connected. To see that p_1, a_1 are connected, we again invoke part (b) of WSPD and claim that p_1, a_1 form a δ-separated pair (A_{p_1}, A_{a_1}), so there is an edge between some point $p_1* \in A_{p_1}$ and some point in $a_1* \in A_{a_1}$. Continuing in this manner, we will eventually decompose the δ-separated pairs into just 2 singletons, i.e., each set in the pair has only one point, say $p_k \in A_{p_k}, q_k \in A_{q_k}$, and these two points by definition are connected by an edge. Retracing our steps, we see that any two points in A_{p_1} are connected, and any two points in A_{a_1} are connected. This means there is a path from p_1 to a_1 through p_1* and a_1*. Thus, p_1, a_1 are connected. A similar argument can be made to show that q_1, b_1 are also connected. From this, we can conclude that there is a path from p_1 to q_1 through a_1 and b_1, which means p_1, q_1 are connected.

Since we showed that any two arbitrary points p_1, q_1 are connected, therefore G is connected.

(b) Given a pair of points p_i, p_j, there is a unique pair (A_k, B_k) that contains this pair. Let A_k have radius r_A, and B_k have radius r_B. Define $r = \max\{r_A, r_B\}$. Then, G contains the edge (a_k, b_k), which has length at most $2r + r/\delta$. Since the only edge between a point in A_k and B_k is (a_k, b_k), therefore a path from p_i to p_j must contain this edge. Thus, we have the following inequality:

$$\begin{aligned} d_G(p_i, p_j) &\leq \|p_i - a_k\| + \|a_k - b_k\| + \|b_k - p_j\| \\ &\leq 2r + (2r + r/\delta) + 2r \\ &= 6r + r/\delta \end{aligned}$$

At the same time, $\|p_i - p_j\|$ is smallest when they are closest together, which is when p_i, p_j are at the edge of A_k, B_k closest to each other, respectively. Thus, we have the following inequality:

$$r/\delta \leq \|p_i - p_j\|$$

With the above two inequalities, we fix ϵ, and we show that:

$$\begin{aligned} d_G(p_i, p_j) &\leq (1 + \epsilon)\|p_i - p_j\| \\ 6r + r/\delta &\leq (1 + \epsilon)(r/\delta) \\ 6\delta + 1 &\leq 1 + \epsilon \\ 6\delta &\leq \epsilon \\ \delta &\leq \epsilon/6 \end{aligned}$$

So, given any ϵ, we just need to choose our δ to be at most $\epsilon/6$ for the first inequality to hold.

Problem 2

(a) The probability of choosing a particular location in T is $1/m$. Given this fact, the probability that a cell in T stores exactly k items is

$$\begin{aligned}
\text{Binomial}(n, k, 1/m) &= \binom{n}{k}(1/m)^k(1-1/m)^{n-k} \\
&\leq (ne/k)^k(1/m)^k(1-1/m)^{n-k} \\
&= (ne/k \cdot 1/m)^k(1-1/m)^n(m/(m-1))^k \\
&= \left(\frac{ne}{k(m-1)}\right)^k(1-1/m)^n
\end{aligned}$$

(b) From Erikson's notes, we see that for a given bin j,

$$P(X_j \geq k) = \binom{n}{k}(1/m)^k$$

Using the approximation from part (a) for the binomial term, we have:

$$\begin{aligned}
P(X_j \geq 2e\ln m) &= \left(\frac{m\ln me}{2e\ln m}\right)^{2e\ln m}(1/m)^{2e\ln m} \\
&= (1/2)^{2e\ln m} = (1/e^{\ln 2})^{2e\ln m} \\
&= \frac{1}{e^{2e(\ln 2)(\ln m)}} = \frac{1}{m^{2e\ln 2}} \\
&= \left(\frac{1}{m^2}\right)^{e\ln 2} \leq \frac{1}{m^2}
\end{aligned}$$

since $1/m^2 < 1$ and $e\ln 2 > 1$. Thus, the probability that a cell in T storing at least $2e\ln m$ items is at most $1/m^2$.

(c) From part (b), we have the probability that a cell in T storing at least $2e\ln m$ items is at most $1/m^2$, then the probability that any cell have at least $2e\ln m$ items is at most:

$$m \times 1/m^2 = 1/m$$

since the events are not independent. Then, the probability that no cell has more than $2e\ln m$ items is at least:

$$1 - 1/m$$

Problem 3

To show that the sum-of-square problem SSQ is NP-complete, we will first show that SSQ$\in$NP, then show that the partition problem PT can be reduced to SSQ.

- Given a set $A = (a_1, \ldots, a_n)$ and integers J, k, and a partition $A_1, \ldots, A_k$, we can verify in polynomial time that $\sum_{i=1}^{k}(\sum_{a \in A_i} a)^2 \leq J$ by adding up every value in each A_i, squaring the sum of each subset, and adding then up. Therefore, SSQ$\in$NP.

- The partition problem is given a set $A = (a_1, \ldots, a_n)$, determine if there is a partition of the n numbers into two disjoint sets A_1, A_2 such that $\sum_{a \in A_1} a = \sum_{a \in A_2} a$. Suppose we have a partition of A into A_1, A_2 such that the sum of each subset is equal to that of the other. To reduce the partition problem to SSQ, let $J = 2(\sum_{a \in A_2} a)^2$, and $k = 2$. Then, the partition problem for set A has a solution if and only if the sum-of-squares problem for set A with $J = 2(\sum_{a \in A_2} a)^2$, and $k = 2$ has a solution.

  Proof. Suppose partition problem for set A has a solution. Then we have disjoint partitions A_1, A_2, where $A_1 + A_2 = A$, such that

$$\sum_{a \in A_1} a = \sum_{a \in A_2} a$$

$$\left(\sum_{a \in A_1} a\right)^2 = \left(\sum_{a \in A_2} a\right)^2$$

$$\left(\sum_{a \in A_1} a\right)^2 + \left(\sum_{a \in A_2} a\right)^2 = 2\left(\sum_{a \in A_2} a\right)^2 \leq J$$

  where each equality implies if and only if. Since we let $k = 2$, the last equality shows that the sum-of-squares problem indeed has a solution. Note this reduction takes polynomial time (in fact, just constant time). □

Since we have shown that SSQ$\in$NP and the reduction from the partition problem, we conclude that SSQ is NP-complete.

Problem 4

To show that the dominating set problem DSP is NP-complete, we will first show that DSP$\in$NP, then show that the vertex cover problem VC can be reduced to DSP.

- Given a graph $G(V, E)$ and a set $S \subseteq V$, we can certainly check if S is a dominating set by checking whether each $u \in V$ is also in S or if u has a neighbor in S. This verification can be done in polynomial time. Therefore, DSP$\in$NP

- To reduce VC, which is known to be NP-complete, to DSP, suppose we have a graph $G_1(V_1, E_1)$ and its vertex cover VC of size k. We construct graph $G_2(V_2, E_2)$ as follows:

  - Copy G_1 into G_2.
  - For each $(u, v) \in E_1$, make a new vertex $z \in V_2$ and new edges $(u, z), (v, z) \in E_2$.

  This graph G_2 certainly can be constructed in polynomial time (in fact, just linear time in $|E|$ after copying G_1 to G_2). Then, VC is the vertex cover of G_1 with size k if and only if it is also a dominating set of G_2 with size at most k.

  Proof. If VC is the vertex cover of G_1, then for each edge $(u, v) \in E_1$, at least one of $u, v \in V_1$. Then in G_2, the same u, v are either in the dominating set or has a neighbor in the dominating set. By construction, every $z \in V_2 - V_1$ must be a neighbor of two vertices where at least one of them is in the dominating set. Since VC has size k, VC is the dominating set of G_2 with size at most k.

  Conversely, Suppose I is the dominating set of G_2 with size at most k. Without loss of generality, suppose $|I| = k$. Then each $u \in V_2$ is either in I or has a neighbor in I. By construction, every $z \in V_2 - V_1$ is a neighbor of exactly two vertices $u, v \in V_1$. Thus, if I were the dominating set of G_2, then each element of I must also be at least one of the two vertices that are neighbors of $z \in V_2 - V_1$. This is equivalent to each element in I must be at least one of each vertex in each edge $(u, v) \in V_1$, which means I is a vertex cover of G_1 with size k. $\square$

Since we have proved that DSP$\in$NP and the reduction from the vertex cover problem, we conclude that DSP is NP-complete.

Problem 5

(a) We can use the bipartite graph algorithm in a previous homework to determine if graph is 2-colorable.

A graph is 2-colorable if the graph has no cycles or if all cycles have a even number of elements. We will use BFS search to determine if the cycles have even number of elements.

Algorithm 1 Determine if graph is bipartite

```
function BFS(node $s$, graph $G(V, E)$)
    for all $u \in V$ do $d(u) = \infty$
    for all $s \in V$ do
        if $s$ is unmarked then
            $d(s) = 0$, mark $s$
            $Q = \emptyset$
            enqueue($s, Q$)
            while $Q \neq \emptyset$ do
                $u$ =dequeue($Q$)
                for all $(u, v) \in E$ do
                    if $d(v) = \infty$ then
                        $d(v) = d(u) + 1$, mark $v$
                        enqueue($v, Q$)
                    else
                        if $d(v) - d(u)$ is even then
                            return "graph is not bipartite"
    return "graph is bipartite"
```

Proof of correctness. If the graph has no cycles, then starting from source s, every other node is in set A and all other nodes are in set B. This implies we can color all nodes in A red and all the nodes in B blue since there is no edge between nodes in A or nodes in B. Similarly, in a cycle with even number of elements, choose any node to begin and place this node in set A. Following the path, the second node will be in set B, as well as every nth node where n is even. On the other hand, if the graph has an odd cycle, it cannot be 2-colroable. To see this, suppose for now odd cycle graphs are 2-colorable. Consider a simple example, a triangle graph with all 3 nodes (say u, v, w) connected by edges. Certainly, this graph has an odd cycle (3 nodes). If we color u red and v blue, then w can be neither red nor blue. Thus we have a contradiction and conclude odd cycle graphs are not 2-colorable.

The algorithm uses BFS to search the entire graph, and will also detect a cycle when it returns to node v and sees $d(v) \neq \infty$ since this means v has already been visited. In this case, determining if $d(v) - d(u)$ is even is sufficient to say graph is not bipartite. To see this, suppose the index of the first element in the cycle is i. If number of elements in the cycle, say k, is odd, then the index of the last element is $i + k - 1$, which is even if i is even or odd if i is odd. Thus the difference between the indices, which is $i + k - 1 - i = k - 1$, which is even. □

Proof of runtime. BFS has $O(|V| + |E|)$ runtime; the algorithm above only add constant work of computing $d(v) - d(u)$ to BFS. Thus the algorithm also has $O(|V| + |E|)$ runtime. □

(b) Algorithm to color G with at most $k + 1$ colors. Begin at any vertex u and run BFS from that vertex. Color u and its neighbors using all distinct colors. For each neighbor v of vertex u, if a neighbor of v, call it w, is not colored, then color w with any color different from colors of neighbors of w. Repeat this until all vertices are colored.

Proof of correctness. The algorithm certainly checks every vertex because BFS correctly visits every vertex. Since each vertex has maximum degree of k, we will never run out of the colors in each step. To see that no neighboring vertices will have the same color, recall that vertices in each subsequent level in the BFS procedure will be colored with a different color than their predecessor. In addition, at each new vertex explored, all its neighbors are checked for color (or no color at all) to ensure the color of the new vertex will be different from all its neighbors. □

Proof of runtime. BFS runs in $O(|V|)$ time. Checking neighbors of w for each node as mentioned in the algorithm takes at most $2|E|$ since number of neighbors is 2 times number of edges. Together, the algorithm runs in $O(|V| + |E|)$. □