CIS 502 - Algorithms

Fall 2014 Short Answers

Problem 1: (20 Points) You are given an array $A[1], \ldots, A[n]$ which contain both negative and positive values. Assume $A[0] = A[n+1] = 0$.

(a) (10 points) Find an interval $[i, j]$ such that $\sum_{t=i}^{j} A[t]$ is maximized. More credit will be provided to a solution which is more efficient. An $O(n^2)$ solution will not get any credit.

(b) (10 points) Find two intervals $[i, j]$ and $[k, \ell]$ where $i \leq j < k \leq \ell$ such that $\sum_{t=i}^{j} A[t] + \sum_{t=k}^{\ell} A[t]$ is maximized. More credit will be provided to a solution which is more efficient. An $O(n^4)$ solution will not get any credit.

Solution: **Part a** Let $B[j]$ be the largest sum of any interval of $1, \ldots j$ ending in j. Now if we include $j-1$ in that interval then we should include the largest interval containing $j-1$. Therefore:

$$B[j] = \max\{B[j-1] + A[j], A[j]\}$$

Initialize $B[0] = 0$ (empty interval or the interval $[0, 0]$. Final answer is $\max_j B[j]$. Running time $O(n)$.

Solution: Part b Create $C[i]$ to be the largest sum of any interval starting from i (switching i for j and $n+1$ for 0 and $i+1$ for $j-1$). Let $C'[i]$ be $\max_{s \geq i} C[i]$ which means the largest interval sum starting afetr i (this can be computed in $O(n)$ time).

Now final answer is $\max_j B[j] + C'[j+1]$. Total time $O(n)$.

Problem 2:(20 Points) You are given a directed graph $G = (V, A)$ with integer capacities c_e. Every vertex v other than s also satisfies the constraint that the sum of the capacities of the edges going out of v is more or equal to the sum of the capacities of the edges coming in to v. Suppose that the maxflow from s to t is f. Prove that the maxflow from t to s is at least f.

Solution: This is similar to the homework question where all edge capacities were 1 and the indegree was equal to the outdegree. We first prove that the sum of the edge capacities of the edge capacities going out of any subset of vertices containg t (and not s) is more than the sum of the edge capacities of the edges coming in.

We then note that max-flow mincut duality and that max flow from s to t is f implies that the sum of the edge capacities of the edges coming in to any set containing t and not s is at least f. Therefore the sum of the edge capacities of the edges leaving any set containing t and not s is at least f. That means the max t–s flow is at least f.

Problem 3: (20 Points) Recall that the minimum spanning tree (MST) is defined as follows: given a weighted graph $G = (V, E, w)$, the weight of any tree is the sum of the weights of the edges in the tree. Our goal is to find the minimum weight tree which contains every vertex in V.

1. (5 points) Professor Xcluster has found himself a large cluster that can perform computations in parallel. He immediately implemented the following round based algorithm for computing the minimum spanning tree in a weighted graph $G = (V, E, w)$: in a round every vertex chooses the smallest weight edge incident to it. After the round is over we merge the connected components into a super vertex and an edge (u, v) becomes the edge $(super(u), super(v))$

where $super(u) \neq super(v)$ are the connected components containing u and v respectively (we ignore loops). We repeat for several rounds till there is one node. Give a **short** proof that the algorithm is correct. What property would the proof rely on?

2. (15 points) Professor Xcluster, found out that instead of computing the minimum, his code only managed to compute the minimum approximately. In particular, if the weight of the correct minimum weight edge incident to any vertex was x the algorithm only guaranteed that the edge output for that vertex was at most $2x$. All the edge weights were non-negative. He correctly believed that he found a 2-approximate minimum spanning tree, but had no proof. Provide a proof that Xcluster found a 2-approximate minimum spanning tree.

   (Hint: Professor 002 suggests considering the new graph G' which has the same vertex and edge sets as the original G but a different set of weights. If an edge e was not in the tree found by Xcluster, its weight remained the same; otherwise (it e was in the tree) then its weight is $w_e/2$. You can assume part (a) for this part.)

Solution: The property is the cut property. For the second part suppose we found a tree T of weight $w(T)$ in part 1. Observe that the tree found in part 1 is the correct minimum spanning tree for G'. Therefore the weight of the MST T' of G is more than the $w(T)/2$.

Problem 4: (20 points) Professor Npsrus wanted to use the cluster to solve his favorite problem of "(Minimum) Spanning Tree with forbidden edge pairs" where we are given a weighted graph $G = (V, E)$ and are given t pairs of edges $P_1, P_2, \ldots, P_t$ such that for any pair $P_r = (e_1(r), e_2(r))$ both the edges $e_1(r)$ and $e_2(r)$ cannot be in the tree simultaneously (but it is fine if the tree contains one or none of them). Show that it is NP hard to decide if there exists a spanning tree with forbidden edge pairs of weight at most K. (Hint: Professor 002 suggests considering the Independent Set problem.)

Solution: Given an IS instance G on n nodes consider the star graph G' with $n+1$ nodes. Node 0 is at the center and has two edges to each vertex i, of weight 0 and 1 respectively, labelled e_i and e'_i respectively. Now if (u, v) is an edge in G then we forbid the pair e_u, e_v. We set $K = n - k$.

Other constructions with paths of length k also exist.

Problem 5: (20 Points) Consider the following problem: we are given a n items from a large universe $[U] = 0, \ldots, U-1$. We do not know which items are provided to us beforehand and there are only r distinct items. We decided to hash every element of the universe, independently of other values, to a value chosen uniformly at random in the range $[0, 1]$ of real numbers.

1. (5 points) Let $t = k/r$ for some integer $k > 0$. What is the best upper bound of the probability that $k/2$ elements have hash value less than t? We are more interested in knowing how you would bound the probability than the exact value. Provide the definition of the random variables you use to get full credit.

2. (5 points) What is the best upper bound of the probability that $2k$ or more elements have hash value greater than t? We are more interested in knowing how you would bound the probability than the exact value. Provide the definition of the random variables you use to get full credit.

3. (10 points) Suppose we knew how to create and store such an hash function. Professor 001 believes that he has an algorithm for estimating r; he wants to iterate over the n items and at any point store only $O(k)$ hash values. At the end of the iteration he wants to output a number r' such that $r/2 \leq r' \leq 2r$ with probability $1 - \frac{2}{n}$. What value of k should he choose and which hash values should he maintain in the iteration?

(Hint: Professor 002 recalls the Markov Inequality for a nonnegative random variable X to be $Pr[X \geq aE[X]] \leq 1/a$ for any $a > 0$. He also recalls the Chebyshec bound to be $Pr[|X - E[X]| \geq aE[X]] \leq Var[X]/(aE[X])^2$ for any $a > 0$. Finally he recalls the Chernoff bounds, where $X_1, X_2, \ldots, X_h$ are independent 0/1 random variables with $X = \sum_i X_i$, $\mu = E[X]$ that $Pr[|X - \mu| \geq a\mu] \leq 2e^{-a^2\mu/3}$ for any $a < 1$.)

Solution: Define X_i to be the even that the hash of element i was less of equal t. This is a 0/1 random event with probability t. The sum $E[\sum_i X_i] = rt = k$ (there are only r distinct elements). We now apply Chernoff bound for part (1) and (2). For (3) we need to set $k = O(\log n)$ and store the smallest k values (or the largest $n - k$).