CSCI 670:
Advanced Analysis of Algorithms Fall 2021

Quiz 2 (Duration: 100 mins)

USC Viterbi
School of Engineering

Name:

Student ID:

Email Address:

Problem 1 (25 points)**.** *A cut $C, V \setminus C$ of an undirected graph $G = (V, E)$ is a partition of vertex set V. We say that a cut C^* is a max-cut if the number of edges between C^* and $V \setminus C^*$ is the maximum possible among all cuts. Design a polynomial time 2-approximation algorithm for the Max-Cut problem. Prove that your algorithm terminates in polynomial time, and its output is 2 approximation.*

Solution 1. *Consider the following algorithm.*

- *Start with an arbitrary cut C.*
- *While there exists a vertex v such that moving v from one side of C to the other increases the number of edges crossing C, move v and update C.*

Notice that in each iteration the cut size is increasing. Since the cut size can be at most $\binom{n}{2}$, it will evantually terminate. In fact, the number of moves made by the algorithm is $O(n^2)$.

Let's assume that the algorithm is terminated. Then, moving any of the vertices does not improve the cut size. Consider an arbitrary vertex v, and WLOG assume that $v \in C^-$ in one side of the cut (C^+ is the other side). Then, we have

$$\#\{(v,u) \in E \ : \ u \in C-\} \leq \#\{(v,u) \in E \ : \ u \in C^+\}.$$

Therefore, for each of the vertices at least half of their neighbors are on the other side. Thus, cut size is at least $\binom{n}{2}/2 \geq OPT/2$.

CSCI 670:
Advanced Analysis of Algorithms Fall 2021

USC Viterbi School of Engineering

Quiz 2 (Duration: 100 mins)

Problem 2 (25 points)**.** *Let $T = (V, E)$ be a tree such that each vertex $v \in V$ has a unique label $\ell_v \in \{1, \dots n\}$. Consider a path $v_1 \to v_2, \cdots \to v_k$, we say that a pair of vertices (v_i, v_j) on the path creates an inversion if $i < j$ and $\ell_{v_i} > \ell_{v_j}$. Design an $O(n \log n)$ algorithm to count the total number of inversion of length k paths of the tree T. (Hint: Observe that we consider each path in both derictions separetely.)*

Solution 2. *First, we make an observation for number of inversions on a path. Let $\ell_1, \dots \ell_k$ be the labels on a length k path. Any pair $i < j$ is an inversion in either in the direction from 1 to k or in the direction from k to 1. Therefore, the number of inversion is $\binom{k}{2} \cdot \#\{\text{length } k \text{ paths}\}$. So, we will apply the algorithm we derived in the homework.*

Assume for a moment we can count the number of length k paths passing through an arbitrary vertex v efficiently. Then, consider the graph once we remove the vertex v. Notice that all remaining paths we want to count lie in smaller connected components. Then, we can count them by using the same procedure recursively. Instead of choosing an arbitrary vertex v, if we choose the median of the tree, then we ensure that subproblems have a size less than half of the original one. So, it remains to design a way to count the number of paths passing through v efficiently.

Algorithm 1 Count length k paths

1: Find the median of the tree, say v.
2: Let $u_1, \dots u_d$ be the neighbors of v.
3: Consider the connected components (subtrees) obtained by removing v. Let T_i be the subtree containing u_i.
4: Count the length k paths in each of the connected components.
5: Define

$$S_i[t] = \#\{\text{lengh } t \text{ paths of } T_i \text{ starting from } u_i\} \quad \textit{and} \quad S[t] = \sum_{i=1}^{d} S_i[t].$$

6: Compute the number of length k paths crossing v:

$$\sum_{i=1}^{d} \sum_{t=1}^{k-1} (S[k-t] - S_i[k-t]) \cdot S_i[k]$$

7: Output the total number obtained from lines 4 and 6

First of all we will discuss that the algorithm is correct and its runtime is $O(n \log n)$. Notice that line 5 can be implemented in linear time. Since in each recursion step we split the problem into subproblems of size at most half of the current problem size, the runtime of the algorithm is $O(n \log n)$. We can prove by induction the algorithm outputs desired number. It is enough to prove that line 6 counts right amount of paths. So, any path of length $t < k$ in the subtree T_i can be merged by a path of size $k-t$ from other subtrees to consist of a path of length k which contains the v. Therefore, Algorithm 1 counts the total number of length k paths.

Let K be the output of the Algorithm 1, then $\frac{K}{2} \cdot \binom{k}{2}$ will be total number of inversion on length k paths.

Problem 3 (25 points)**.** *A function $f(x) : [0, 1] \to \mathbb{R}$ is called unimodal if there exists a real $x^* \in [0, 1]$ such that*

- *$f(y_2) < f(y_1)$ for all $0 \leq y_2 < y_1 \leq x^*$*
- *$f(y_1) > f(y_2)$ for all $x^* \leq y_1 < y_2 \leq 1$.*

Design an $O\left(\log \frac{1}{\epsilon}\right)$ algorithm to find an approximate solution x' to global maximum x with at most ϵ additive error, i.e. $|x' - x| \leq \epsilon$.

Solution 3. *We will implement an algorithm similar to binary search. However its decision mechanism will be different. The following algorithm returns an approximate optimal value of the function f.*

Algorithm 2 Search(f, begin,end)

if end $-$ begin $< \epsilon$ **then**
  Output: begin
end if
Let $x_1 = \text{begin} + \frac{\text{begin} + \text{end}}{3}$.
Let $x_2 = \text{end} - \frac{\text{begin} + \text{end}}{3}$.
if $f(x_1) \leq f(x_2)$ **then**
  Output: Search(f, x_1, end)
else
  Output: Search(f, begin, x_2)
end if

First, we show that the output is an ϵ approximation to the optimal solution. Observe that each recursive step is called by an interval containing the optimal solution.

- *$f(x_1) \leq f(x_2)$. Observe that maximum element can not be in the interval of* begin $: x_1$ *since the array is unimodal. Thus, it is enough to search between x_1 and* end*.*
- *$f(x_1) > f(x_2)$. This case is symmetric to the previous one.*

Thus, in each iteration we branch with the correct interval to search the maximum element and the size of the problem is decreases. Since the size of the problem is monotonicially decreasing, the algorithm will eventually find an interval of length at most ϵ which contains the maximum element. Any point on the resulting interval will be ϵ approximation to the problem.

Observe that the length of the interval at step k is $\left(\frac{2}{3}\right)^k$. Then,

$$\left(\frac{2}{3}\right)^k \leq \epsilon \iff k \cdot \log(2/3) \leq \log(\epsilon)$$

Therefore, $k = O(\log(1/\epsilon))$ is enough to get ϵ-approximation.

Problem 4 (25 points)**.** *Let $a_0, a_1, \ldots a_{n-1}$ and $b_0, b_1, \ldots b_{m-1}$ be two sequences of reals. A convolution operator on two sequences defined as follows*

$$\mathbf{c} = \mathbf{a} * \mathbf{b} \qquad \text{where } c_k = \sum_{i=0}^{m-1} a_{k-i} b_i \text{ for all } k \in \{0, \ldots n+m-2\}$$

where $a_j = 0$ for any $j < 0$ or $j > n-1$. Design an $O(n \log n)$ algorithm for computing convolution $\mathbf{a} * \mathbf{b}$ of two sequences.

Solution 4. *Consider the polynomials $p(x) = p_0 + p_1 x + \cdots + p_{n-1} x^{n-1}$ and $q(x) = q_0 + q_1 x + \cdots + q_{n-1} x^{n-1}$, where $p_i = a_i$ for all i; $q_i = b_i$ when $i \leq m-1$; and $q_i = 0$ when $i \geq m$. Let $r(x) = r_0 + r_1 x + \cdots + r_{2n-2} x^{2n-2}$ be the product of $p(x)$ and $q(x)$. We have that*

$$r_k = \sum_{i=0}^{k} p_i q_{k-i}, \quad k = 0, \ldots, n+m-2$$

where $p_j = 0$ when $j \geq n$ and $q_j = 0$ when $j \geq m$. Then, we have

$$r_k = \sum_{i=k-m+1}^{k} p_i \cdot q_{k-i} = \sum_{i=0}^{m-1} p_{k-i} \cdot q_i = \sum_{i=0}^{k} a_{k-i} b_i.$$

So, we set c_k to the coefficient r_k of x^k in $p \cdot q$. Summing up, to find c_k (the coefficients of the convolution) we need to find the coefficients of $r(x)$, which can be done in $O(n \log n)$ time using the discrete FFT algorithm.