CSE 5500 Algorithms. Fall 2018

Homework 2 Solutions

1. The coefficients of the polynomial are given by $\binom{n}{i}$, $i = 0, 1, \ldots, n$.
   Since $\binom{n}{i} = \binom{n}{i-1}\ (n-i+1)/i$, the coefficients can be computed in time $O(n)$.
   FFT can be used to multiply two nth degree polynomials in $O(n \log n)$ time. We can compute the coefficients of $(1+x)^n$ by multiplying $(1+x)^{n/2}$ and $(1+x)^{n/2}$. If $T(n)$ is the time needed to compute $(1+x)^n$, then, $T(n) = T(n/2) + O(n \log n)$, which solves to $O(n \log n)$.

2. Let A be a *Toeplitz* matrix and B be an $n \times 1$ vector. Let's consider the multiplication of the lower triangular part of A (including the main diagonal elements) with B.
   Let the elements of A be the following:
   $a_{n,n} = a_{n-1,n-1} = a_{n-2,n-2} = \ldots = a_{2,2} = a_{1,1} = a_1$
   $a_{n,n-1} = a_{n-1,n-2} = a_{n-2,n-3} = \ldots = a_{2,1} = a_2$
   $a_{n,n-2} = a_{n-1,n-3} = a_{n-2,n-4} = \ldots = a_{3,1} = a_3$
   $\vdots$
   $a_{n,1} = a_n$
   Let the elements of B be the following:
   $b_{1,1} = b_1, b_{2,1} = b_2, \ldots, b_{n,1} = b_n$
   Multiplication of the lower triangular part of A with B gives the following:
   $$\begin{bmatrix} a_1 b_1 \\ a_2 b_1 + a_1 b_2 \\ a_3 b_1 + a_2 b_2 + a_1 b_3 \\ \vdots \end{bmatrix}$$
   We can notice that the above is nothing but the multiplication of two polynomials $(a_1 + a_2 x + a_3 x^2 + \ldots)$ and $(b_1 + b_2 x + b_3 x^2 + \ldots)$.
   Since the plolynomials can be multiplied in $O(n \log n)$ time, the matrices can also be multiplied in $O(n \log n)$ time. Multiplication of the upper triangular elements of A with B is symmetrical to the above and it would not affect the asymptotic complexity.

3. Using Taylor series expansion for $f(.)$,
   $f(a+x) = f(a) + x f^1(a) + \frac{x^2}{2!} f^2(a) + \ldots + \frac{x^n}{n!} f^n(a)$
   where $f^i(x)$ stands for the ith derivative of $f(x)$. Let $F(x)$ denote $f(a+x)$. Evaluate $F(x)$ at the nth roots of unity. This can be done in $O(n \log^2 n)$ time as was mentioned in class (see Section 9.5 in the text – An nth degree polynomial can be evaluated at n arbitrary points in $O(n \log^2 n)$ time). Then, use inverse FFT to compute the coefficients of $F(x)$. This can be done in $O(n \log n)$ time. Once the coefficients of $F(x)$ are known, it is easy to determine the derivatives.
   Run time $= O(n \log^2 n) + O(n \log n) + O(n) = O(n \log^2 n)$.

   Another solution: Let $f(x) = a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0$. Then, $f'(x) = n a_n x^{n-1} +$

$(n-1)a_{n-1}x^{n-2}+\cdots+2a_2x+a_1$; $f''(x) = n(n-1)a_nx^{n-2}+(n-1)(n-2)a_{n-1}x^{n-3}+\cdots+2a_2$; $f'''(x) = n(n-1)(n-2)a_nx^{n-3}+(n-1)(n-2)(n-3)a_{n-1}x^{n-4}+\cdots+6a_3$; and so on. Now consider the following two polynomilas: $A(x) = (n!a_n)x^{n-1} + ((n-1)!a_{n-1})x^{n-2} + \cdots + (2!)a_2x + a_1$ and $B(x) = \frac{a^{n-1}}{(n-1)!} + \frac{a^{n-2}}{(n-2)!}x + \cdots + \frac{a^2}{2!}x^{n-3} + ax^{n-2} + x^{n-1}$. Compute the product of $A(x)$ and $B(x)$ in $O(n \log n)$ time. We can obtain the required derivatives from the coefficients of this product. The total run time is $O(n \log n)$. However note that the coefficients of $A(x)$ could become very large creating practical difficulties.

4. Let C be a cycle with k vertices. Let the edge e of C with the maximum weight be a part of a minimum-cost spanning tree of G. There exists atleast one edge, e', of C which is not a part of the minimum-cost spanning tree (k vertices can be connected by $k-1$ edges and C contains k edges). By replacing e in the spanning tree with e', we can obtain a new spanning tree whose cost will be less than that of the original minimum-cost spanning tree. This is a contradiction and hence, the edge with the maximum weight of C can not be part of a minimum-cost spanning tree of G.

5. Let $X = x_1, x_2, x_3, \ldots, x_n$ and $Y = y_1, y_2, y_3, \ldots, y_m$.
Let $L(i,j)$ represent the length of the longest common subsequence between $x_1, x_2, \ldots, x_i$ and $y_1, y_2, \ldots, y_j$ such that the common subsequence ends at x_i and y_j in X and Y respectively. Assuming that $L[i-1, j-1]$ has already been calculated, compute $L[i,j]$ as follows:
if $x[i] = y[j]$ then $L[i,j] = L[i-1, j-1] + 1$
else $L[i,j] = 0$.
Compute $L[i,j]$ for $0 \le i \le n$; $0 \le j \le m$. Scan through all these values and pick the largest $L[i,j]$. Computing $L[i,j]$ from $L[i-1, j-1]$ takes constant time. Thus the total run time is $O(nm)$.

6. (a) $C(i,j) = min_{i \le k \le j}\{C(i,k) + C(k+1,j) + D(i-1)D(k)D(j)\}$
   (b) for $i = 1$ to r do
   for $j = i+1$ to r do
   for $k = i$ to j do
   $C[i,j] = min(C[i,j], C[i,k] + C[k+1,j] + D(i-1)D(k)D(j)$

7. We can verify all the properties that a flow function is required to satisfy. Let $F = \alpha f_1 + (1-\alpha)f_2$. Then, i) **Capacity constraint:** For all $u, v \in V$ we have: $F(u,v) = \alpha f_1(u,v) + (1-\alpha)f_2(u,v) \le \alpha c(u,v) + (1-\alpha)c(u,v) \le c(u,v)$; ii) **Skew symmetry:** For all $u, v \in V$, $F(u,v) = \alpha f_1(u,v) + (1-\alpha)f_2(u,v) = -\alpha f_1(v,u) - (1-\alpha)f_2(v,u) = -F(v,u)$; iii) **Flow conservation:** For all $u \in V - \{s,t\}$, $\sum_{v \in V} F(u,v) = \sum_{v \in V} \alpha f_1(u,v) + (1-\alpha)f_2(u,v) = \alpha \sum_{v \in V} f_1(u,v) + (1-\alpha)\sum_{v \in V} f_2(u,v) = 0$.

8. Going through the steps of Ford-Fulkerson method we realize that the value of maximum flow is 19.