CIS 502 - Algorithms

Fall 2015 Homework 4

Problem 1. *Problem 4, page 247–248, of textbook.*

Solution: There are two solutions below. Both illustrate the power of **reductions**, i.e., **trying** to solve an unknown problem using known problems. We will keep revisiting reductions. The first solution shows that we can convert the problem into a polynomial multiplication problem. The second shows that we can convert it to computing convolutions. Both are correct – in the second one we use polynomial multiplications to actually solve convolutions – so the two solutions are same. In general remember the principle that **unless asked to directly solve/reprove something shown in class**, you can use any fact/algorithm used in class and simply say as "shown in class."

However the above applies **only** to things proved in class, and if there is an excercise problem in the book which was not discussed in class then you cannot say "as the excercise in the book".

Solution two is cleaner, but solution one is more first principles, i.e., how do we "typically" derive these things in the first place. After the derivation is complete, we often simplify things, and that is the easier way to get to the second solution. But all this being said, some folks will find it easier to think directly about convolutions and some will prefer the polynomial representation to guide them. Both are legit directions.

Solution One: (The reasoning/proof of correctness.) We observe that:

- Directions matter. If $i < j$ then a positive charge at i pushes the charge towards $+\infty$. Otherwise if $i > j$ then the positive charge at i pushes the charge towards $-\infty$.
- The force F_j at location j due to i is q_j times $\frac{q_i}{(i-j)^2}$ times direction of $i \to j$.
- Forces add up.. The force F_j at location j is q_j times $\phi_j = \sum_{i;i\neq j} \frac{q_i}{(i-j)^2} dir(i \to j)$.

So we will consider evaluating $\phi_j = \sum_{i;i\neq j} \frac{q_i}{(i-j)^2} dir(i \to j)$ for all j differently. If we can compute all ϕ_j then a simple $O(n)$ computation of $\phi_j \cdot q_j$ will suffice – provided we get the directions right. Now the "field" or $\phi_j = \sum_i a_{ij} q_i$ where a_{ij} is the field/effect determined by a unit charge at i (and we are summing up over all i). Observe that

$$a_{ij} = \begin{cases} \frac{1}{(i-j)^2} & \text{when } i < j \\ 0 & \text{when } i = j \\ -\frac{1}{(i-j)^2} & \text{when } i > j \end{cases}$$

where the convention is that if the force is towards to $+\infty$ part of the line then it is +ve and otherwise the force is −ve. Note that if q_i, q_j have opposite signs then the sign is reversed since $F_j = q_j\phi_j = q_j \sum_i a_{ij} q_i$. *Note that $A(x) = \sum_j a_{ij} x^{j-i}$ is a polynomial which is independent of i or j. This polynomial has degree $-(n-1)$ to $n-1$ (the minimum and maximum values of $j-i$).*

Consider the polynomial $\Phi(x) = \sum_j \phi_j x^j$.

$$\Phi(x) = \sum_j \phi_j x^j = \sum_j \sum_i a_{ij} q_i x^j = \sum_i q_i x^i \sum_j a_{ij} x^{j-i} = \sum_i q_i x^i A(x)$$

Therefore if we define $B(x) = \sum_i q_i x^i$ then $\Phi(x) = A(x)B(x)$ which is a polynomial multiplication problem! We have one small issue, the polynomial $A(x)$ contains powers of x which are negative – to avoid this we can evaluate $x^n\Phi(x) = (x^n A(x))\, B(x)$ and now that problem is solved. The coefficient of x^{j+n} in $x^n\Phi(x)$ is ϕ_j.

Algorithm: We construct the polynomials $(x^n A(x))$ and $B(x)$ in time $O(n)$ each. The maximum degree of $(x^n A(x)), B(x), x^n\Phi(x)$ is at most $2n$. We now use the algorithm for multiplying polynomials for an n' which is a power of 2 and $n'/2 < n \le n'$.

Running time: This algorithm runs in time $O(n' \log n') = O(n \log n)$. Once we have ϕ_j for all j, we can compute the forces in $O(n)$ time.

Solution Two: As shown in the book (page 237) the definition of convolution is

$$c_k = \sum_{(i,j):i+j=k} a_i b_j$$

We will use the fact that we can compute convolutions over domains of size n in time $O(n \log n)$. The problem ask to compute the convolution over vectors of size $2n-1$.

$$\mathbf{a} = (q_1, q_2, \ldots, q_i, \ldots q_n, 0, \ldots, 0)$$
$$\mathbf{b} = \left(-\frac{1}{(n-1)^2}, \ldots, -\frac{1}{4}, -1, 0, 1, \frac{1}{4}, \ldots, \frac{1}{(n-1)^2}\right)$$

The padding by extra 0's make sure that the $i+j=k$ uses all the terms. We now need to show that the

$$c_{n+k} = \sum_i \sum_{j:i+j=n+k} q_i b_j = \sum_{i<k} \frac{q_i}{(k-i)^2} - \sum_{i>k} \frac{q_i}{(k-i)^2}$$

The above follows because b_j is $1/(j-n)^2$ when $j > n+1$ and if $n+k = i+j$ then $j-n = k-i$. Likewise b_j is $-1/(n-j)^2$ when $j < n+1$ and in that case we get $1/(k-i)^2$. This proves the correctness of the algorithm. Running time: $O(n \log n)$.

Problem 2. *Problem 7, page 248–249, of textbook.*

Solution: Let row 0 be at the bottom and column 0 at the left.

First, find the minimum value on the boundary – this takes at most $4n$ probes. Suppose this is node $(0, i)$ (you can rotate the solution below for the other case).

Now if the value at $(1, i)$ is larger than the value at $(0, i)$ then we have a local minima. So we know that the value at $(1, i)$ is less than that of $(0, i)$. Since $(0, i)$ is the minimum over all the boundary we know $G(i, 1)$ is lower than all values in the boundary.

Now divide G into four halves and find the minimum value of the $2n-5$ new points in the two divider lines. Let this value be at (a, b). We have two cases:

- Suppose that the value at (a, b) is more than the value of $(0, i)$. Then if we focus on the quadrant which contains $(1, i)$ we will maintain the invariant that $G(i, 1)$ is lower than all values in the boundary of this smaller box (of size $n/2$).
- Suppose that the value at (a, b) is less than the value of $(0, i)$. Then one side of (a, b) must have a value smaller than the value at (a, b) – otherwise we have a local minima. Let that node which has smaller value than the value at (a, b) be p. Now consider the quadrant which contains p – the value of p is lower than all values in the boundary of this smaller box (of size $n/2$).

If we proceed as the above we will eventually have a situation of 4×4 grid where the value of a point p is less than the all values in the boundary. We can now find a local minima in 16 steps — we start moving from p towards the direction where the value decreases. If we have no moves, we have a local minima. If we move, we maintain the invariant that the value at the node is less than the values on the boundary. We cannot visit a node a twice. Therefore we must find a local minima in at most 16 steps.

Overall time: Let $T[n]$ be the maximum number of probes for an $n \times n$ grid after we have established the invariant (value of some internal node less than the entire boundary). Then we make $2n-5$ probes to find the minimum and (maybe) probe the two sides of that minimum value. Therefore

$$T[n] = (2n-3) + T[n/2]$$

Note that we recurse on only a single quadrant. The solution is $T[n] = O(n)$. The total number of probes is $4n - 4 + T[n]$ (counting the boundary).

Problem 3. *Problem 5, page 316–317, of textbook.*

Solution: Let $A[i]$ be the best (maximum) quality segmentation of $y_1, \ldots, y_i$. We are interested in computing $A[n]$.

$$A[i+1] = \max_{j \le i} A[j] + \mathit{quality}(y_{j+1}, \ldots, y_{i+1})$$

We are trying all ways of instantiating the last word $y_1, \ldots, y_{i+1}$, and it ccan be $y_{j+1}, \ldots, y_{i+1}$.
Running time $O(n^2)$ – we are evaluating $O(n)$ subproblems each of which take $O(n)$ time. Initial condition: $A[0] = 0$.

Problem 4. *Problem 6, page 317–318, of textbook.*

Solution: Let $A[i]$ be the best (minimum) quality printing of $w_1, \ldots, w_i$. We are interested in computing $A[n]$.

$$A[i] = \max_{j \le i, \sum_{j'=j}^{i}(c_{j'}+1) - 1 \le L} A[j-1] + \left(L - \sum_{j'=j}^{i}(c_{j'}+1) - 1 \right)^2$$

We are trying all possible lines that can be packed in the last line, which can be $c_j, \ldots, c_i$.
Running time $O(n^2)$ – we are evaluating $O(n)$ subproblems each of which take $O(n)$ time or does it? Note that if we compute the partial sums $\sum_{j'=1}^{i} c_j$ on the side *then* we can evaluate the expression in the reccurrence in $O(1)$ time. Initial condition: $A[0] = 0$.