CIS 502

Yezheng Li

September 30, 2015

1 Homework 4

Question (5-4) *You have been working with some physicists who need to study, as part of their experimental design, the interactions among large numbers of very small charged particles. Basically, their setup works as follows. They have an inert lattice structure, and they use this for placing charged particles at regular spacing along a straight line. Thus we can model their structure as consisting of the points* $\{1, 2, \ldots, n\}$ *on the real line; and at each of these points* j*, they have a particle with charge* q_j*. (Each charge can be either positive or negative.)*

They want to study the total force on each particle, by measuring it and then comparing it to a computational prediction. This computational part is where they need your help. The total net force on particle j*, by coulomb's Law, is equal to*

$$F_j \doteq \sum_{i<j} \frac{Cq_iq_j}{(j-i)^2} - \sum_{i>j} \frac{Cq_iq_j}{(j-i)^2},$$

They have written the following simple program to compute F_j *for all* j*:*

Code 1 *For $j = 1, 2, \ldots, n$;*

1. *Initialize $F_j \leftarrow 0$;*
2. *For $i = 1, 2, \ldots, n$*
   (a) *If $i < j$ then Add $\frac{Cq_iq_j}{(j-i)^2}$ to F_j;*
   (b) *Else if $i > j$ then Add $-\frac{Cq_iq_j}{(j-i)^2}$ to F_j.*
   (c) *Endif*
3. *Output F_j.*

It is not hard to analyze the running time of this program: each invocation of the inner loo, over i, takes $O(n)$ time, and this inner loop is invoked $O(n)$ times total, so the overall running time is $O(n^2)$.

The trouble is, for the large values of n , the program takes several minutes to run. On the other hand, their experimental setup is optimized so that they can throw down n particles, perform the measurements, and be ready to handle n more articles within a few seconds. So they had really like it if there were a way to compute all the forces F_j much more quickly, so as to keep up with the rate of the experiment.

Help them out by designing an algorithm that computes all the forces F_j in $O(n \log n)$ time.

Answer *Define*

$$Q(x) \doteq \sum_{j=0}^{n-1} q_{j+1} x^j,$$

$$R(x) \doteq x^{n-1} \left(-\sum_{j=1}^{n-1} \frac{x^{-j}}{j^2} + \sum_{j=1}^{n-1} \frac{x^j}{j^2} \right),$$

and

$$\hat{Q}(y) \doteq \sum_{s=0}^{2n-1} Q(\omega^{-s}) y^s, \hat{R}(y) \doteq \sum_{j=0}^{2n-1} R(\omega^{-s}) y^s, \text{where } 1 + \omega + \ldots + \omega^{2n-1} = 0.$$

[1]

[1] here we keep along with section **conversion and fast Fourier transform**. On the

Code 2 *1. Use fast fourier transform to calculate convolution of P, Q:*

(a) calculate $\hat{Q}(y)$ from $Q(x)$ –$O(n\log n)$;

(b) calculate $\hat{R}(y)$ from $R(x)$ –$O(n\log n)$;

(c) calculate $\hat{P}(y) \doteq \sum_{s=1}^{2n-1} Q(\omega^{-s})R(\omega^{-s})y^s$ – $O(n)$;

(d) calculate $P(x) = \sum_{j=1}^{2n-1} p_j x^j \doteq \sum_{j=0}^{2n-1} \frac{\hat{P}(\omega^j)}{2n} x^j$ –$O(n\log n)$.

Notice now

$$\begin{aligned}
p_{n-1} &= -\frac{q_n}{(n-1)^2} - \ldots - \frac{q_2}{1^2} \\
&= -\sum_{i>1}^{n} \frac{q_i}{(1-i)^2}, \\
p_n &= -\frac{q_n}{(n-2)^2} - \ldots - \frac{q_3}{(1^2} + \frac{q_1^2}{1^2} \\
&= \frac{q_1^2}{1^2} - \sum_{i>2}^{n} \frac{q_i}{(2-i)^2}, \\
p_{n+1} &= -\frac{q_n}{(n-3)^2} - \ldots - \frac{q_4}{1^2} + \frac{q_2^2}{1^2} + \frac{q_1^2}{2^2} \\
&= \sum_{i<3}^{n} \frac{q_i}{(3-i)^2} - \sum_{i>3}^{n} \frac{q_i}{(3-i)^2}, \\
\textit{(in all) } p_{n-2+j} &= \sum_{i<j}^{n} \frac{q_i}{(j-i)^2} - \sum_{i>j}^{n} \frac{q_i}{(j-i)^2} = \frac{F_j}{q_j},
\end{aligned}$$

2. Compute $F_j \leftarrow p_{n-2+j}q_j, j = 1, \ldots, n$ – $O(n)$.

Running time in all $O(n\log n)$.

---

other hand, we can also choose ω as $1 + \omega + \ldots + \omega^{2n-2} = 0$ and correspondingly

$$\hat{Q}(y) \doteq \sum_{s=0}^{2n-2} Q(\omega^{-s})y^s, \hat{R}(y) \doteq \sum_{j=0}^{2n-2} R(\omega^{-s})y^s, \text{where } 1 + \omega + \ldots + \omega^{2n-2} = 0.$$

Question (5-7) *Suppose now that you are given an $n \times n$ grid graph G.*

We use some of the terminology of the previous question. Again, each node v is labeled by a real number x_v; you may assume that all these labels are distinct. Show how to find a local minimum of G using only $O(n)$ probes to the nodes of G. (Note that G has n^2 nodes.)

Answer (refers to [3]) *Denote the set of nodes on the* **border** *of the grid G by*

$$\begin{aligned} B \quad &\doteq \{(1,k) \in G : k = 1 \ldots, n\} \cup \{(k,1) \in G : k = 1 \ldots, n\} \\ &\cup \{(n,k) \in G : k = 1 \ldots, n\} \cup \{(k,n) \in G : k = 1 \ldots, n\}. \end{aligned}$$

Say that G has **Property** *$(*)$*

If it contains a node $v \notin B$ that is adjacent to a node in B and satisfies $x_v < x_p, \forall p \in B$.

Note that in a grid G with Property $(*)$, the global minimum *does not occur on the border B (since the global minimum is no larger than v, which is smaller than B) — hence G has at least one local minimum that does not occur on the border. We call such a local minimum an* **internal local minimum**.

Code 3

1. *Find the node p on the border B of minimum value;*
2. *If p is a corner node return v;*
3. *(Denote has a unique neighbor of p not on B by v)*
   If $x_p < x_v$, then return v;
4. *(G satisfies Property $(*)$ since v is smaller than every node on B)*
   a recursive algorithm:
   (a) *Denote the union of the nodes in the middle row and middle column of G, not counting the nodes on the border by*

$$\begin{aligned} C \doteq \quad &\{(n/2,k) \in G : k = 1, \ldots, n\} \\ &\cup \{(k,n/2) \in G : k = 1, \ldots, n\} \setminus B. \end{aligned}$$

   Deleting $B \cup C$ from G divides up G into four sub-grids.

(b) *(let T be all nodes adjacent to S.)*
Find the node $u = arg \min_{u \in S \cup T} x_u$ – $O(n)$ probes;
(Notice that $u \notin B$, since $v \in S \cup T$ and $v \prec B$.)

(c) *If $u \in C$, then return u (since all of the neighbors of u are in $S \cup T$, and u is smaller than all of them.);*

(d) *otherwise $u \in T$. Let G' be the sub-grid containing u, together with the portions of S that border it. (Notice G' satisfies Property $(*)$, since u is adjacent to the border of G' and is smaller than all nodes on the border of G'.)*
run the recursive algorithm on G' (since G' has an internal local minimum, which is also an internal local minimum of G.)

If $T(n)$ denotes the number of probes needed by the algorithm to find an internal local minimum in an $n \times n$ grid, we have the recurrence $T(n) = O(n) + T(n/2)$, which solves to $T(n) = O(n)$.

In all, the running time is $O(n)$.

Question (6-5)

Answer *1. Denote $BQ(x_1 \dots x_m)$, the best quality of a long string $x_1 \dots x_n$ with x_i characters;*

2. Final answer: $BQ(x_1 \dots x_n)$ with characters $c_1, \dots, c_n$ inputed;

3. $BQ(x_1 \dots x_n) \doteq \min_{1 \le m \le n-1} \{BQ(x_1 \dots x_m) + BQ(x_{m+1} \dots x_n)\}$;

4. Initialization I don't know how to do it;

5. Running time: considering $x_i, i = 1, \dots, m$ where $m \le n$ where $x_i \in C$, the set of all characters (say, $\{'a', 'b', 'c' \dots, 'z'\}$). Since the number of input entries are at most $\#C^n$, I think running time is $O(\#C^n)$.

Question (6-6)

Answer *1. Denote $V[c_1, \dots, c_m]$ the minimal of the sum of the squares of the slacks of all lines consisting of m words each with number of characters $c_i, i = 1, \dots, m$;*

2. *Final answer:* $V[c_1, \ldots, c_n]$ *with input* $c_1, \ldots, c_n$*;*

3. $V[c_1, \ldots, c_n] \doteq \min_{1 \le m \le n-1} \{V[c_1, \ldots, c_m] + V[c_{m+1}, \ldots, c_n]\}$*;*

4. *Initialization* $V[c_1, \ldots, c_m] = \begin{cases} \left[L - \sum_{i=1}^{m}(c_i + 1) - 1\right]^2, & L \ge \sum_{i=1}^{m}(c_i + 1) - 1; \\ \infty, & L < \sum_{i=1}^{m}(c_i + 1) - 1. \end{cases}$

5. *Running time: Considering* $c_i, i = 1, \ldots, m$ *where* $m \le n$*. If* $c_k > L$ *for some* $k \in [m]$*, it can be easily detected that* $V[] = \infty$ *rather than any other value; in normal case,* $c_i \in [L], i = 1, \ldots, m$*. Since the number of input entries are at most* n^L*, I think running time is* $O(n^L)$*.*

References

[1] en.wikipedia.org.

[2] Jon Kleinberg, va Tardos, *Algorithm Design*, Addison-Wesley: 2005-3-26, 864 pages

[3] solcornell.tex.