CS521 Fall 2011 \ Extra Credit Assignment

Ariel Stolerman

Page 110, Question 4-6 – Monge arrays:

a.

Let A be a $m \times n$ matrix. We will prove that:

(1) The Monge array property: $\forall i,j,k,l\ s.t. 1 \le i < k \le m, 1 \le j < l \le n$: $A[i,j] + A[k,l] \le A[i,l] + A[k,j] \Leftrightarrow$

(2) $\forall i = 1,2,\dots,m-1\ \&\ j = 1,2,\dots,n-1$: $A[i,j] + A[i+1,j+1] \le A[i,j+1] + A[i+1,j]$

Proof (by induction):

(1) ⇒ (2):

If (1) applies on A that is a $m \times n$ matrix, then $A[i,j] + A[k,l] \le A[i,l] + A[k,j]$ is satisfied for all $1 \le i < k \le m$ and $1 \le j < l \le n$, so it is specifically satisfied for $k = i+1, l = j+1$, which immediately derives (2).

(2) ⇒ (1):

As suggested we will proof separately for rows and columns. The proof is similar so we show only for rows.

We will prove that for $1 \le i < k \le m$: $A[i,j] + A[k,j+1] \le A[i,j+1] + A[k,j]$.

In the base case $k = i+1$ and we get that exactly (2): $A[i,j] + A[i+1,j+1] \le A[i,j+1] + A[i+1,j]$.

For the inductive step we assume correctness for all $1 \le i < p < k$: $A[i,j] + A[p,j+1] \le A[i,j+1] + A[p,j]$. Specifically, it applies for $p = k-1$, and we get:

$$A[i,j] + A[k-1,j+1] \le A[i,j+1] + A[k-1,j] \Rightarrow A[i,j] - A[i,j+1] \le \underbrace{A[k-1,j] - A[k-1,j+1]}_{(*)}$$

From assuming (2) we know:

$$A[k-1,j] + A[k,j+1] \le A[k-1,j+1] + A[k,j] \Rightarrow \underbrace{A[k-1,j] - A[k-1,j+1]}_{(*)} \le A[k,j] - A[k,j+1]$$

$$\Rightarrow A[i,j] - A[i,j+1] \le A[k,j] - A[k,j+1] \Rightarrow \boxed{\forall 1 \le i < k \le m: A[i,j] + A[k,j+1] \le A[i,j+1] + A[k,j]}$$, as required.

Doing the same for columns will derive that $\boxed{\forall 1 \le j < l \le n: A[i,j] + A[i+1,l] \le A[i,l] + A[i+1,j]}$.

Combined together we get the Monge array property (1), as required.

b.

By applying (a) we know we can only check all 2×2 sub-matrices for the property, and that will be sufficient. By checking that we find that if we increase $A_{1,3}$ from 22 to 24, the array becomes Monge.

c.

Let $f(i)$ be the index of the column containing the leftmost element of row i. We will prove that $f(1) \leq f(2) \leq \cdots \leq f(m)$ for any $m \times n$ Monge array.

We assume by contradiction that for some $i \in \{1,2,\ldots,m-1\}$: $f(i) > f(i+1)$. The minimum elements of rows i and $i+1$ are $A[i,f(i)], A[i+1,f(i+1)]$ by definition of f, thus $A[i,j] \geq A[i,f(i)]$ for any $j \neq f(i)$ and $A[i+1,j] \geq A[i+1,f(i+1)]$ for any $j \neq f(i+1)$. Specifically: $A[i,f(i+1)] \geq A[i,f(i)]$ & $A[i+1,f(i)] \geq A[i+1,f(i+1)] \Rightarrow A[i,f(i+1)] + A[i+1,f(i)] \geq A[i,f(i)] + A[i+1,f(i+1)]$

Since we assumed $f(i) > f(i+1)$, we will change the notations to fit with the original Monge array property such that $k = i+1, f(i+1) = j, f(i) = l$ and we get $A[i,j] + A[k,l] \geq A[i,l] + A[k,j]$, which contradicts the Monge array property.

d.

Given the indices of the columns with the leftmost minimum per each even row, we find the column index of the leftmost minimum of all odd rows as follows:

By (c) we know that $\forall i: f(i-1) \leq f(i) \leq f(i+1)$, so for all odd i's we only need to look between column $f(i-1)$ and $f(i+1)$, which are given from the even rows' minimum calculation earlier, in order to find $f(i)$. The total is:

- For $f(1)$: $f(2) - 0 + 1$ elements to inspect.
- For $f(3)$: $f(4) - f(2) + 1$ elements to inspect.
- …
- Without loss of generality, assume m is even, so for $f(m-1)$: $f(m) - f(m-2) + 1$ elements to inspect.

(If m is odd, the last would be: $f(m)$: $n - f(m-1) + 1$).

Note that summing all together we get a telescopic sum plus $1 \times \sim m/2$ (depending if the number of rows is even or odd), which is a total of: $f(m) + \frac{m}{2}$ for an even m and $n + \left\lceil \frac{m}{2} \right\rceil$ for an odd m. In both cases that's $O(n) + O(m) = O(n+m)$, as required.

e.

Let A be a $m \times n$ matrix input to the algorithm described in (d). Each recursive call gets an input matrix with the same number of columns n and half the number of rows of the level above it. Given what we proved in (d), the recurrence is: $T(m) = T(m/2) + O(n) + O(m)$ (omitting floor signs). Since we divide m by 2 at each recursive call, we will end up with $\lg m$ recursive calls. At each call, the n-part stays $O(n)$ since the number of columns of the input doesn't change from call to call, so the total cost for the n-part is $O(n \lg m)$. But the number of rows is divided by 2, so the total m-part cost is $\sum_{k=1}^{\lg m} m/2^k = m \cdot \frac{(1/2)^{\lg m - 1} - 1}{1/2 - 1} = m \cdot \frac{1 - 2 \cdot (1/2)^{\lg m}}{1 - 1/2} = m\left(2 - 4m^{\lg 1/2}\right) = 2m - 4 = O(m)$. Therefore the total cost of the m-part is $O(m)$. That concludes to a total cost of $O(m + n \lg m)$ for the entire algorithm described in (d).

Page 188, problem 7-5 – Median-of-3 partition:

a.

Here is a formula for p_i as a function of n for $i = 2,3,\dots,n-1$ (for $i = 1, n: p_i = 0$):

$$p_i(n) = \underbrace{3\frac{1}{n}}_{\substack{chance\ the\ i^{th}\ o.s.\ is \\ one\ of\ the\ 3\ chosen}} \cdot \underbrace{2\frac{i-1}{n-1}}_{\substack{chance\ one\ of\ the\ two \\ left\ is < A[i]}} \cdot \underbrace{\frac{n-i}{n-2}}_{\substack{chance\ the\ last\ left \\ is > A[i]}} = \frac{6(i-1)(n-i)}{n(n-1)(n-2)}$$

b.

In the original implementation of Randomized-Quicksort, the pivot is chosen by random, thus for an array of size n the probability for the lower median $i = \left\lfloor \frac{n+1}{2} \right\rfloor$ to be chosen is $\frac{1}{n}$.

Now, assuming $n \to \infty$ (thus omitting floor notations and taking $\frac{n-1}{2}$ as the median index) the probability of the median to be chosen using the median-of-3 method is $\frac{6\left(\frac{n+1}{2}-1\right)\left(n-\frac{n+1}{2}\right)}{n(n-1)(n-2)} = \frac{6\left(\frac{n+1-2}{2}\right)\left(\frac{2n-n-1}{2}\right)}{n(n-1)(n-2)} = \frac{3(n-1)^2}{2n(n-1)(n-2)} = \frac{3(n-1)}{2n(n-2)} = \frac{3n-3}{2n^2-4n}$

Now we will calculate the ratio between the improved and the original and look at it asymptotically:

$\lim_{n\to\infty} \frac{3n-3}{2n^2-4n} / \frac{1}{n} = \lim_{n\to\infty} \frac{3n^2-3n}{2n^2-4n} = \lim_{n\to\infty} \frac{3-\frac{3}{n}}{2-\frac{4}{n}} = \boxed{\frac{3}{2}} \Rightarrow$ we increased the chance of selecting the median by 50%.

c.

For a good split that is defined by choosing $\frac{n}{3} \le i \le \frac{2n}{3}$, the probability to get such split is:

$$\int_{\frac{n}{3}}^{\frac{2n}{3}} \frac{6(i-1)(n-i)}{n(n-1)(n-2)} \partial i = \frac{6}{n(n-1)(n-2)} \int_{\frac{n}{3}}^{\frac{2n}{3}} -i^2 + (n+1)i - n \partial i = \frac{6}{n(n-1)(n-2)} \cdot \left[-\frac{i^3}{3} + \frac{(n+1)i^2}{2} - ni \right]_{\frac{n}{3}}^{\frac{2n}{3}} =$$

$$\frac{6}{n(n-1)(n-2)} \cdot \left[-\frac{8n^3}{81} + \frac{4n^2(n+1)}{18} - \frac{2n^2}{3} + \frac{n^3}{81} - \frac{n^2(n+1)}{18} + \frac{n^2}{3} \right] = \frac{6}{n(n-1)(n-2)} \cdot \left[-\frac{7n^3}{81} + \frac{n^2(n+1)}{6} - \frac{n^2}{3} \right] =$$

$$\frac{1}{(n-1)(n-2)} \cdot \left[-\frac{14n^2}{27} + n^2 + n - 2n \right] = \frac{1}{(n-1)(n-2)} \cdot \left(\frac{13}{27} n^2 - n \right) = \frac{13n^2 - 27n}{27(n^2 - 3n + 2)} = \frac{13n^2 - 27n}{27n^2 - 81n + 54}$$

When $n \to \infty$ we get the following probability:

$$\lim_{n\to\infty} \frac{13n^2 - 27n}{27n^2 - 81n + 54} = \lim_{n\to\infty} \frac{13 - \frac{27}{n}}{27 - \frac{81}{n} + \frac{54}{n}} = \frac{13}{27}$$

If we used the original algorithm, the chance of getting a good split as defined above is $\frac{1}{3}$. Therefore the ratio is:

$\frac{13}{27} / \frac{1}{3} = \frac{13}{27} \cdot 3 = \frac{13}{9} = 1\frac{4}{9} \cong 1.44$, thus by using the median-of-three we increased the chances of getting a good split as defined above by $\sim 44\%$.

d.

The best case partition is picking the median at each iteration ($T(n) = 2T(n/2) + \Theta(n)$), which derives $\Omega(n \lg n)$ running time for quicksort. Therefore any other method of choosing the pivot, including the median-of-three, will not affect the running time lower bound by more than a constant, concluding to $\Omega(n \lg n)$ anyway.