CS 521: Data Structures and Algorithms I
Extra Credit

Dustin Ingram

November 28, 2011

1. **(4-6) Solution:**

   (a) **Base Case:** A 2×2 Monge Array:

   $$A[i,j] + A[i+1,j+1] \leq A[i,j+1] + A[i+1,j]$$
   $$A[1,1] + A[2,2] \leq A[1,2] + A[2,1]$$

   Inductive Step (Rows): A $m \times 2$ array, $m \geq 2$:
   If $m = 2$, $i + 1 = m = 2$, $i = 1$, and the base case holds:

   $$A[i,j] + A[i+1,j+1] \leq A[i,j+1] + A[i+1,j]$$
   $$A[i,1] + A[m,2] \leq A[i,2] + A[m,1]$$
   $$A[1,1] + A[2,2] \leq A[1,2] + A[2,1]$$

   Then, adding a new row so that $i = m$ and $i + 1 = m + 1$:

   $$A[i,j] + A[i+1,j+1] \leq A[i,j+1] + A[i+1,j]$$
   $$A[m,1] + A[m+1,2] \leq A[m,2] + A[m+1,1]$$

   The equalities can be transformed:

   $$A[i,1] + A[m,2] \leq A[i,2] + A[m,1] \Rightarrow A[i,1] - A[i,2] \leq A[m,1] - A[m,2]$$
   $$A[m,1] + A[m+1,2] \leq A[m,2] + A[m+1,1] \Rightarrow A[m,1] - A[m,2] \leq A[m+1,1] - A[m+1,2]$$

   Combining these:

   $$A[i,1] - A[i,2] \leq A[m,1] - A[m,2] \leq A[m+1,1] - A[m+1,2]$$

   Therefore:

   $$A[i,1] - A[i,2] \leq A[m+1,1] - A[m+1,2]$$
   $$A[i,1] + A[m+1,2] \leq A[m+1,1] + A[i,2]$$

   Inductive Step (Columns): A $2 \times n$ array, $n \geq 2$:
   If $n = 2$, $j + 1 = n = 2$, $j = 1$, and the base case holds:

   $$A[1,j] + A[2,n] \leq A[2,j] + A[1,n]$$

Then, adding a new column so that $j = n$ and $j + 1 = n + 1$:

$$A[1, n] + A[2, n + 1] \leq A[2, n] + A[1, n + 1]$$

The equalities can be transformed and combined as in the first step, therefore:

$$A[1, j] + A[2, n + 1] \leq A[1, n + 1] + A[2, j]$$

Inductive Assumption: A $m \times n$ array, $m < 2$, $n < 2$:

$$A[m, n] + A[m + 1, n + 1] \leq A[m, n + 1] + A[m + 1, n]$$

(b) Using the inductive assumption above, it reveals that the bolded sub-array is invalid:

$$\begin{pmatrix} 37 & \mathbf{23} & \mathbf{22} & 32 \\ 21 & \mathbf{6} & \mathbf{7} & 10 \\ 53 & 34 & 30 & 31 \\ 32 & 13 & 9 & 6 \\ 43 & 21 & 15 & 8 \end{pmatrix}$$

Therefore, one element within it must be changed. By assigning the following variables to this subarray:

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix}$$

The goal will be to satisfy

$$a + d \leq b + c$$

The following inequalities can be generated based on each element's local Monge array dependencies:

$$\begin{aligned} 37 + 6 \leq a + 21 &\Rightarrow 22 \leq a \\ b + 10 \leq 7 + 32 &\Rightarrow b \leq 29 \\ 37 + c \leq 23 + 21 &\Rightarrow c \leq 7 \\ 21 + 34 \leq c + 53 &\Rightarrow 2 \leq c \\ c + 30 \leq 7 + 34 &\Rightarrow c \leq 11 \\ 6 + 30 \leq d + 34 &\Rightarrow 2 \leq d \\ d + 31 \leq 10 + 30 &\Rightarrow d \leq 9 \\ 22 + 10 \leq 32 + d &\Rightarrow 0 \leq d \end{aligned}$$

Which can be reduced to the following external dependencies:

$$\begin{aligned} 22 \leq &a \\ &b \leq 29 \\ 2 \leq &c \leq 7 \\ 2 \leq &d \leq 9 \end{aligned}$$

The inequalities which show how the variable must be changed within the bolded sub-array are as follows:

$$\begin{aligned} a + 7 \leq 22 + 6 &\Rightarrow a \leq 21 \\ 23 + 7 \leq b + 6 &\Rightarrow 24 \leq b \\ 23 + 7 \leq 22 + c &\Rightarrow 8 \leq c \\ 23 + d \leq 22 + 6 &\Rightarrow d \leq 5 \end{aligned}$$

It is clear that a and c cannot be modified to hold all dependencies. The resulting inequalities show the possible ranges for b and d:

$$24 \leq b \leq 29$$
$$2 \leq d \leq 5$$

Therefore making $d = 5$ results in the following Monge array:

$$\begin{pmatrix} 37 & 23 & 22 & 32 \\ 21 & 6 & \mathbf{5} & 10 \\ 53 & 34 & 30 & 31 \\ 32 & 13 & 9 & 6 \\ 43 & 21 & 15 & 8 \end{pmatrix}$$

(c) Take an array A such that $f(i) > f(i+1)$; for example, where the minimum of the i-th row is found at A[i,j] and the minimum of the $i+1$-th row is found at A[i+1, j-1]. This means that the sum

$$A[i, j] + A[i + 1, j - 1]$$

is the lowest possible pair of values for the i-th and $i + 1$-th rows. However, for the sub-array containing these values to be Monge, the following must be true:

$$A[i, j - 1] + A[i + 1, j] \leq A[i, j] + A[i + 1, j - 1]$$

Which requires that there exist another pair of values less than the pair of minimums. Therefore, $f(1) \leq f(2) \leq \cdots \leq f(m)$ for any $m \times n$ Monge array.

(d) For a square $n \times n$ Monge array A, the left-most minimum for each odd-numbered row (given the leftmost minimum of the preceding even-numbered row) can be be in only one of two locations: if the leftmost minimum of the preceding even-numbered row is at $A[i, j]$, the leftmost minimum of the following odd-numbered row must be at $A[i+1, j]$ or $A[i+1, j+1]$ to satisfy the condition that $f(i) \leq f(i+1)$. Since there are $n/2$ odd-numbered rows to search, this results in a complexity of $O(2 * (n/2)) = O(n)$. However, for a non-square $n \times m$ Monge array, there can be at most m additional elements between $f(1)$ and $f(m)$, therefore the complexity is at most $O(n + m)$.

2. **(7-5) Solution:**

(a) When partitioning n elements at position i, the number of elements before and after the partition are $i-1$ and $n-i$, respectively. This results in $(i-1)(n-1)$ possible three-element subsets which will include pivot element at i. The total number of possible three-element subsets from a set of n elements is simply $\binom{n}{3}$. Therefore, the probability that any given three-element subset will contain the pivot element is the ratio of subsets containing the pivot element to the number of possible subsets, or:

$$\frac{(i-1)(n-i)}{\binom{n}{3}} = \frac{(i-1)(n-i)}{\frac{n!}{3!(n-3)!}} = \frac{(i-1)(n-i)}{\frac{n(n-1)(n-2)}{6}}$$

(b) We can assume that i is even, therefore let $i = \frac{n}{2}$. The resulting probability is:

$$\begin{aligned}
\frac{(\frac{n}{2}-1)(n-\frac{n}{2})}{\frac{n(n-1)(n-2)}{6}} &= \frac{6(\frac{n^2}{2}-\frac{n^2}{4}-n+\frac{n}{2})}{(n-1)(n^2-2n)} \\
&= \frac{6(\frac{n^2}{4}-\frac{n}{2})}{(n-1)(n^2-2n)} \\
&= \frac{\frac{6}{4}(n^2-2n)}{(n-1)(n^2-2n)} \\
&= \frac{3}{2(n-1)}
\end{aligned}$$

The ratio of this probability to the probability of the ordinary implementation, $\frac{1}{n}$, is simply:

$$\frac{\frac{3}{2(n-1)}}{\frac{1}{n}} = \frac{3n}{2n-2}$$

As $n \to \infty$, this becomes $\frac{3}{2}$, or a $1.5\times$ improvement.

(c) The probability of getting a 'good' split with the ordinary implementation is $\frac{1}{3}$... not sure how to approximate the sum by an integral.

(d) In the best-case scenario, the recursion tree of QUICKSORT will be perfectly divided, with height $\lg n$ and n leaves, resulting in a $\Omega(n \lg n)$ running time, so the median-of-three method cannot improve upon this scenario.