CSOR 4246, Fall 2018 **Brief solutions for hw1t**

1. **Solution to problem 1**

   (a) Straightforward proof by induction.

   (b) This routine requires n additions and n multiplications. Let $n = 2^k$ and consider the polynomial $p(x) = x^n$. This polynomial can be evaluated at x using $\log n$ multiplications only by repeatedly squaring x.

2. **Solution to Problem 2**

   (a) Split the n-dimensional vector $\mathbf{v}$ into vectors $\mathbf{v}^t$ and $\mathbf{v}^b$ which consist of the top and bottom $n/2$ entries of $\mathbf{v}$, respectively.
   Similarly, split the n-dimensional vector $H_k\mathbf{v}$ into $(H_k\mathbf{v})^t$ and $(H_k\mathbf{v})^b$. Then

$$
\begin{aligned}
(H_k\mathbf{v})^t &= H_{k-1}\mathbf{v}^t + H_{k-1}\mathbf{v}^b = H_{k-1}(\mathbf{v}^t + \mathbf{v}^b) \\
(H_k\mathbf{v})^b &= H_{k-1}\mathbf{v}^t - H_{k-1}\mathbf{v}^b = H_{k-1}(\mathbf{v}^t - \mathbf{v}^b)
\end{aligned}
$$

   To compute $H_k\mathbf{v}$ first compute $\mathbf{v}^t + \mathbf{v}^b$ and $\mathbf{v}^t - \mathbf{v}^b$ and then compute the products $H_{k-1}(\mathbf{v}^t + \mathbf{v}^b)$ and $H_{k-1}(\mathbf{v}^t - \mathbf{v}^b)$ recursively.
   Running time: Let $T(n)$ be the time to multiply the $n \times n$ matrix H_k by an n-dimensional vector $\mathbf{v}$. Then $T(n) = cn + 2T(n/2) = O(n \log n)$.

3. **Solution to Problem 3** (Inductive Proof)

   (a) Base case: For $n = 1$, Pr[sample is the i-th item] $= \frac{1}{k} = 1$.

   (b) Hypothesis: Assume that for some $k \geq 1$, the probability of the sample being any of the k elements is $\frac{1}{k}$.

   (c) Step: We will show the statement for $k + 1$.

$$
\begin{aligned}
&\text{Pr[the sample is the } (k+1)\text{-st item]} = \frac{1}{k+1} \\
&\text{Pr[the sample is a different item from the } (k+1)\text{-st]} = 1 - \frac{1}{k+1} = \frac{k}{k+1} \\
&\text{Pr[the sample is the } i\text{-th item]} = \frac{1}{k} \cdot \frac{k}{k+1} = \frac{1}{k+1}, \text{for } i = 1, \ldots, k
\end{aligned}
$$

   Therefore, the probability that any item is the sample is $\frac{1}{k+1}$.

   (b) The sample is the i^{th} item with probability =

$$
\begin{cases}
(\frac{1}{2})^{k-i+1} & i = 2, \ldots, k \\
(\frac{1}{2})^{k-1} & i = 1
\end{cases}
$$

4. **Solution to Problem 4**

   (a) Use standard algorithm for matrix multiplication to compute AB in $\Theta(n^3)$. Compare with C in $O(n^2)$.

(b) Consider the vector $\mathbf{v} = M\mathbf{x}$. Let v_i be the i-th entry of $\mathbf{v}$. For any fixed $1 \le i \le n$, we have

$$\Pr[\mathbf{v} \neq \mathbf{0}] \le \Pr[v_i \neq 0].$$

So it suffices to prove that

$$\Pr[v_i \neq 0] \le \frac{1}{2}.$$

Since M is non-zero, there is at least one entry in M that is not equal to 0; w.l.o.g., say entry $M_{ij} \neq 0$. Since v_i is given by the dot product of the i-th row of M with the vector $\mathbf{x}$, we have

$$v_i = \sum_{k=1}^{n} M_{ik} \cdot x_k = M_{ij} \cdot x_j + \sum_{\substack{k=1 \\ k \neq j}}^{n} M_{ik} \cdot x_k$$

It follows that v_i is 0 if and only if

$$M_{ij} x_j = -\sum_{\substack{k=1 \\ k \neq j}}^{n} M_{ik} x_k$$

Suppose that all the x_k's have been set randomly and independently to $\{0, 1\}$ with probability $1/2$, **except** for x_j.

- If the right hand side above equals 0, then x_j must be set to 0 for v_i to equal 0. This happens with probability $1/2$.
- If the right hand side above does not equal zero, then certainly x_j should not be set to 0 for v_i to equal to 0. Again this happens with probability $1/2$.

Hence $\Pr[v_i = 0] \le \frac{1}{2}$ in either case.

(c) Set $M = AB - C$.

- If $AB = C$, then M is the all-zeros matrix. Hence for any $\mathbf{x}$, $\Pr[M\mathbf{x} = \mathbf{0}] = 1$.
- If $AB \neq C$, then M is a non-zero matrix. By part (b),

$$\Pr[AB\mathbf{x} = C\mathbf{x}] = \Pr[M\mathbf{x} = 0] \le \frac{1}{2}.$$

Hence the randomized test for checking whether $AB = C$ is as follows:

Algorithm 1

`VerifyMatrixProduct`(A, B, C, n, k)

1: **for** iteration i from 1 to k **do**
2:     Generate a random $\mathbf{x}$ as in part (b)
3:     Compute $C\mathbf{x}$, $B\mathbf{x}$, $A(B\mathbf{x})$
4:     Compute $\mathbf{v}_i = AB\mathbf{x} - C\mathbf{x}$
5:     **if** $\mathbf{v}_i \neq \mathbf{0}$ **then**
6:         `return` **no**
7:     **end if**
8: **end for**
9: `return` **yes**   // that is, return **yes** iff $\mathbf{v}_i = \mathbf{0}$ for every $1 \le i \le k$

Fix an i. Line 2 inside the for loop takes $O(n)$ time; line 3 takes $O(n^2)$ time if we first compute $C\mathbf{x}$, $B\mathbf{x}$ and finally $A(B\mathbf{x})$; lines 4 and 5 take time $O(n)$. Hence, the running time of one iteration of the randomized algorithm is $O(n^2)$. Hence the total running time is $O(kn^2)$.

Success probability of the randomized test:

- If $AB = C$, then $\mathbf{v}_i$ is always $\mathbf{0}$. Hence the algorithm *succeeds* —that is, it always returns the correct answer—with probability 1.
- If $AB \neq C$, then the algorithm *fails* —that is, outputs the wrong answer— with probability at most $\left(\frac{1}{2}\right)^k$. Thus the algorithm succeeds with probability at least $1 - \left(\frac{1}{2}\right)^k$.

1. Solution to recommended exercise 1

f	g	O	o	Ω	ω	Θ
$\log^5 n$	$10 \log^3 n$	n	n	y	y	n
$n^2 \log(2n)$	$n \log n$	n	n	y	y	n
$\sqrt{\log n}$	$\log \log n$	n	n	y	y	n
$n^2 + n^{1/3}$	$n^2 \log n + n^{5/2}$	y	y	n	n	n
$\sqrt{n} + 1500$	$n^{1/3} + \log n$	n	n	y	y	n
$\frac{3^n}{n^2}$	$2^n \log n$	n	n	y	y	n
$n^{\log n}$	2^n	y	y	n	n	n
2^n	$\frac{3^n}{n^{\log n}}$	y	y	n	n	n
n^n	$n!$	n	n	y	y	n
$\log n^n$	$\log n!$	y	n	y	n	y

2. Solution to recommended exercise 2

   Give tight asymptotic bounds for the following recurrences.

   - According to master theorem, $a = 4$, $b = 2$, $k = 2$. Thus $a = b^k$ and $T(n) = O(n^k \log n)$, hence

   $$T(n) = O(n^2 \log n)$$

   - According to master theorem, $a = 8$, $b = 2$, $k = 3$. Thus $a = b^k$ and $T(n) = O(n^k \log n)$, hence

   $$T(n) = O(n^3 \log n)$$

   - According to master theorem, $a = 11$, $b = 4$, $k = 2$. Thus $a < b^k$ and $T(n) = O(n^k)$, hence

   $$T(n) = O(n^2)$$

   - According to master theorem, $a = 7$, $b = 3$, $k = 1$. Thus $a > b^k$ and $T(n) = O(n^{\log_b a})$, hence

   $$T(n) = O(n^{\log_3 7})$$

3. Solution to recommended exercise 3

   The recurrence for this algorithm is

   $$T(n) = 3T(2n/3) + \Theta(1) = \Theta(n^{\log_{3/2} 3}) = \Theta(n^{\frac{\log 3}{\log 1.5}}) = \omega(n^2)$$

   Hence both insertion sort and merge Sort are faster than this algorithm.