CS60003 Algorithm Design and Analysis, Autumn 2010–11

Class test 1

Maximum marks: 40 | Time: 07–08–09–10 | Duration: $1 + \epsilon$ hour

---

Roll no: ______________ **Name:** ______________________________________

[*Write your answers in the question paper itself. Be brief and precise. Answer all questions.*]

In both the following exercises, you solve the same computational problem which goes like this. You are given n probabilities $p_1, p_2, \ldots, p_n$ (so each $p_i \in [0,1]$). You are also given an integer k in the range $0 \leqslant k \leqslant n$ (so $k = O(n)$). Assume that p_i is the probability of obtaining a head in a random toss of a coin C_i. One toss is made of each of the coins $C_1, C_2, \ldots, C_n$ in that order. Your task is to propose efficient algorithms to compute the probability $P(n,k)$ of obtaining exactly k heads in these n tosses. Of course, in addition to n and k, the value of $P(n,k)$ depends also on the probabilities $p_1, p_2, \ldots, p_n$. For simplicity, we use the simplified notation $P(n,k)$ to actually stand for $P(n, k, p_1, p_2, \ldots, p_n)$.

Example: Let $n = 3$, $p_1 = \frac{1}{3}$, $p_2 = \frac{1}{2}$, $p_3 = \frac{3}{4}$, and $k = 2$. Denote a head by H and a tail by T. All possible outcomes of three tosses with exactly two heads are HHT, HTH and THH. The probability to be calculated is, therefore, $P(3,2) = \frac{1}{3}\times\frac{1}{2}\times(1-\frac{3}{4})+\frac{1}{3}\times(1-\frac{1}{2})\times\frac{3}{4}+(1-\frac{1}{3})\times\frac{1}{2}\times\frac{3}{4} = \frac{1}{24}+\frac{3}{24}+\frac{6}{24} = \frac{10}{24} = \frac{5}{12}$.

Suppose that we do arithmetic on floating-point numbers of a fixed size (like `double` in C), so the cost of adding, subtracting or multiplying two floating-point values is always $\Theta(1)$. In particular, the probabilities $p_1, p_2, \ldots, p_n$ are supplied as floating-point values (not as rational numbers as in the above example).

If $k = n/2$, there are $\binom{n}{n/2} \geqslant 2^{n/2} = (\sqrt{2})^n$ outcomes with exactly k heads. Enumerating all possibilities leads to fully exponential running time. Better algorithms are needed to achieve polynomial running times.

1. First, design an $O(n^2)$-time dynamic-programming algorithm to compute $P(n,k)$. Use the values $P(i,j)$ to stand for the probability of obtaining exactly j heads in the tosses of $C_1, C_2, \ldots, C_i$.

(a) For $i \geqslant 1$, express $P(i,j)$ in terms of $P(i-1, j-1)$ and $P(i-1, j)$. Give brief justification. **(5)**

(b) Supply conditions to terminate the recursive definition of $P(i,j)$. Give brief justification. **(5)**

(c) Convert the above formulas to an $\mathrm{O}(n^2)$-time dynamic-programming algorithm to calculate $P(n, k)$. **(5)**

(d) Justify that your algorithm runs in $\mathrm{O}(n^2)$ time. **(5)**

2. Now, design an $\mathrm{O}(n \log^2 n)$-time divide-and-conquer (top-down) algorithm for computing $P(n, k)$. Denote by $Q(i, j, k)$ the probability of obtaining exactly k heads in tosses of $C_i, C_{i+1}, \ldots, C_j$. (We have $P(n, k) = Q(1, n, k)$.) Also, denote by $F_{i,j}(x)$ the polynomial

$$F_{i,j}(x) = \sum_{k \geqslant 0} Q(i, j, k) x^k.$$

$Q(i, j, k) = 0$ for $k > j - i + 1$, so $F_{i,j}(x)$ is indeed a polynomial (not a non-terminating power series). If we can compute the polynomial $F_{i,j}(x)$, we output its coefficient of x^k as $Q(i, j, k)$. In particular, $P(n, k)$ is the coefficient of x^k in $F_{1,n}(x)$. So it suffices to compute $F_{1,n}(x)$.

(a) Basis case: Let $i \in \{1, 2, \ldots, n\}$. Write the expression for $F_{i,i}(x)$ with justification. **(5)**

(b) Induction: Let $1 \leqslant i \leqslant m < j \leqslant n$. Prove that $F_{i,j}(x) = F_{i,m}(x) F_{m+1,j}(x)$. **(5)**

(c) Propose an $\mathrm{O}(n \log^2 n)$-time divide-and-conquer algorithm to compute $F_{1,n}(x)$. **(5)**

(d) Prove that your algorithm in Part 2(c) runs in $\mathrm{O}(n \log^2 n)$ time. (You may use without proof the result that the solution of the recurrence $T(n) = 2T(n/2) + \mathrm{O}(n \log n)$ is $T(n) = \mathrm{O}(n \log^2 n)$. For a proof, look at the solution of Exercise 1 in the Mid-Semester Test of Autumn 2008.) **(5)**

(**Remark:** Never ever underestimate the power of top-down programming.)

[You may also use this space for continuation of answers. Give pointers from Pages 1–4.]

[You may also use this space for continuation of answers. Give pointers from Pages 1–4.]