CMPSC 465 Data Structures & Algorithms
Fall 2024 Mingfu Shao and Debarati Das

WS 11

November 4

1. We will consider how much Huffman coding can compress a file F of m characters taken from an alphabet of $n = 2^k$ characters $x_0, x_1, \ldots, x_{n-1}$ (each character appears at least once).

   (a) Let $S(F)$ represents the number of bits it takes to store F without using Huffman coding (i.e., using the same number of bits for each character). Find an expression (formula) for $S(F)$ in terms of m and n.

   (b) Let $H(F)$ represents the number of bits used in the optimal Huffman coding of F. We define the efficiency $E(F)$ of a Huffman coding on F as $E(F) := S(F)/H(F)$. For each m and n describe a file F for which $E(F)$ is as small as possible.

   (c) For each m and n describe a file F for which $E(F)$ is as large as possible. How does the largest possible efficiency increase as a function of n? Give you answer in big-O notation.

   solution

   (a) As the alphabet size is n, each character can be encoded with $\log n$ bits. Following $|F| = m$, the total number of required bits to store F is $O(m \log n)$.

   (b) The efficiency is smallest (worst) when all characters appear with equal frequency. Notice here Huffman coding does not provide any better estimation than the one discussed above. the efficiency is ≈ 1.

   (c) Let F be $x_0, x_1, \ldots, x_{n-2}$ followed by $m-(n-1)$ instances of x_{n-1}. This file has efficiency

   $$\frac{m \log n}{(m-(n-1)) \cdot 1 + (n-1) \cdot \log n} \approx \frac{m \log n}{(m-(n-1)) \cdot 1} \approx \log n$$

   So the efficiency is $O(\log n)$.

2. A server has n customers waiting to be served. Customer i requires t_i minutes to be served. If, for example, the customers were served in the order $t_1, t_2, t_3, \ldots$, then the ith customer would wait for $t_1 + t_2 + \cdots + t_i$ minutes.

   We want to minimize the total waiting time

   $$T = \sum_{i=1}^{n} (\text{ time spent waiting by customer } i)$$

   Given the list of t_i, give an efficient algorithm for computing the optimal order in which to process the customers.

   Solution: We simply proceed by a greedy strategy, by sorting the customers in the increasing order of service times and servicing them in this order. The running time is $O(n \log n)$.

To prove the correctness, for any ordering of the customers, let $s(j)$ denote the jth customer in the ordering. Then

$$T = \sum_{i=1}^{n} \sum_{j=1}^{i-1} t_{s(j)} = \sum_{i=1}^{n} (n-1) t_{s(i)}$$

For any ordering, if $t_{s(i)} > t_{s(j)}$ for $i < j$ then swapping the positions of the two customers gives a better ordering. Since, we can generate all possible orderings by swaps, an ordering which has the property that $t_{s(1)} \leq \cdots \leq t_{s(n)}$ must be the global optimum. However, this is exactly the ordering we output.