UM-SJTU JOINT INSTITUTE
VE477 Introduction to Algorithms

Homework 1

Li Yong 517370910222

September 20, 2020

Ex.1 Hash tables

1. n slots, n keys are equiprobably hashed to each slot, hence the probability that a key hashed to the certain slot is uniform, which is equal to $\frac{1}{n}$.
   Hence according to the binomial distribution, the probability P_k that exactly k keys hash to a same slot is

$$\left(\frac{1}{n}\right)^k \left(1-\frac{1}{n}\right)^{n-k} \binom{n}{k}$$

2. Considering the event X_i that slot i with the most keys to have exactly k keys, i.e., exactly k keys hash to slot $i \wedge$ less than k keys hash to other slots. Hence

$$P(X_i) \leq P_k$$

   The probability P'_k for the slot with the most keys to have exactly k keys

$$P'_k = \sum_{i=0}^{n} P(X_i) \leq nP_k$$

3.

$$\begin{aligned}
P_k &= \left(\frac{1}{n}\right)^k \left(1-\frac{1}{n}\right)^{n-k} \binom{n}{k} \\
&\leq \left(\frac{1}{n}\right)^k \frac{n!}{k!(n-k)!} \\
&\leq \left(\frac{1}{n}\right)^k \frac{n^k}{k!} \\
&= \frac{1}{k!} \\
&\approx \frac{1}{\sqrt{2\pi k}(\frac{k}{e})^k} \\
&\leq \frac{e^k}{k^k}
\end{aligned}$$

4.

$$\begin{aligned}
&P'_k < 1/n^2 \\
&P_k = \frac{e^k}{k^k} < 1/n^3 \\
&\frac{k^k}{e^k} > n^3 \\
&k(\log k - \log e) > 3\log n \\
&c\log n/\log\log n(\log(c\log n/\log\log n) - \log e) > 3\log n \\
&c\log n/\log\log n(\log c + \log\log n - \log\log\log n - \log e) > 3\log n \\
&c(1 + \frac{\log c - \log\log\log n - \log e}{\log\log n}) > 3
\end{aligned}$$

Considering $f(x) = 1 - \frac{\log n}{n}$, $f'(x) = \frac{\log x - \frac{1}{\ln 2}}{x^2}$

$$f'(x) = \frac{\log x - \frac{1}{\ln 2}}{x^2} = 0 \Rightarrow x = e$$

$$f(x)_{max} = f(e) = 1 - \frac{\log e}{e} = 0.469$$

Then

$$c(1 + \frac{\log c - \log\log\log n - \log e}{\log\log n}) = c(1 + 0.469) > 3$$

$$c > 4.407$$

5.

$$\begin{aligned}
E[M] &= \sum_{k=1}^{n} k \cdot Pr(M = k) \\
&= \sum_{k=\lceil \frac{c\log n}{\log\log n} \rceil}^{n} k \cdot Pr(M = k) + \sum_{k=1}^{\lceil \frac{c\log n}{\log\log n} \rceil - 1} k \cdot Pr(M = k) \\
&\leq n \sum_{k=\lceil \frac{c\log n}{\log\log n} \rceil}^{n} Pr(M = k) + \frac{c\log n}{\log\log n} \sum_{k=1}^{\lceil \frac{c\log n}{\log\log n} \rceil - 1} Pr(M = k) \\
&= Pr\left(M > \frac{c\log n}{\log\log n}\right) n + Pr\left(M \leq \frac{c\log n}{\log\log n}\right) \frac{c\log n}{\log\log n}
\end{aligned}$$

Ex.2 Minimum spanning tree

Algorithm 1 Minimum spanning tree

Input: A minimum spanning tree T of $G =< V, E >$, the decreased edge $E(u, v)$

Output: A new minimum spanning tree T' of $G =< V, E >$

1: $T' \leftarrow T + E$
2: $E' \leftarrow$ the edge with the largest weight in the circle in T' formed by E
3: **if** E'.weight $> E$.weight **then**
4: $\quad T' \leftarrow T' - E'$
5: **else**
6: $\quad T' \leftarrow T$
7: **end if**
8: **return** T'

Ex.3 Simple algorithm

Algorithm 2 Sum of n-bit integers

1. **Input:** Array X, Y with n elements, which store two n-bits integers
   Output: Array Z with $n+1$ elements, which store the sum of the two integers

```
1: count←0;
2: for i = n to 1 do
3:     Z[i] = (X[i] + Y[i] + count) / 10;
4:     count = (X[i] + Y[i]) % 10;
5: end for
6: Z[n+1] = count / 10;
7: return Z
```

Algorithm 3 mult(x, y)

2. (a) **Input:** Two integers x, y
   Output: Product of x and y

```
1: if x = 0 or y = 0 then
2:     return 0
3: end if
4: return mult(2x, ⌊y/2⌋) + x · (y mod 2)
```

   (b) Considering the bit operation of x and y, we can regard $\lfloor y/2 \rfloor$ as $y >> 1$, then (y mod 2) get the last bit of y, which will be left by $>>$ operation. The last bit of y multipled by $2x$ can be regard as shift the last bit left then multipled by x. In other words, the algorithm seperate y in binary bits and calulate the product of each bit and x, then combinate the results by bit operation.

Ex.4 Problem

The minimum number of races necessary is 7.

Seperate 25 horses into 5 groups of 5 horse. Race these 5 groups. Then race the fastest horse of each group to obtain the fastest horse among the 25 horses. Then pick the 2nd and 3rd fastest horses in the two race that the fastest horse joined, also the 2nd fastest horse in the group race that the 2nd horse in the latest race joined. Then race the picked horses. Top 2 of the race will be the 2nd and 3rd fastest horse among the 25 horses.

Ex.5 Critical thinking

1. First of all, both of the algorithms can be failed in some cases. However, compared with each other, "largest first" is much better. Here I list two reasons:

   i. If both of the algorithms give solutions, "smallest first"'s solution is probably less optimal than "largest first"'s. Unless all the numbers in S are consumed, the sum of some smaller numbers that consist of the solution can be substituted by the larger number.

ii. Also, I found an article that proved that under the condition that $S = \{s \mid s = c^k, k \in \mathbb{N}\}$, "largest first" greedy algorithm must have the optimal solution.

2. If m is a power of 2, then it will be an extreme deficient case. When we operate the hash function $H(k) = k \bmod m$ in bit operation, we noticed that when $m = 2^n$, $H(k)$ will be the lowest n bits of k, i.e., it wastes part of the bits. So that we would like to choose an m not too close from a power of 2 to avoid the deficiency.
   Also, we perfer a prime for it can reduce the number of common factors between k and m. If k shares a common factor with m, k will be hased to a bucket that is a multiple of the common factor, so that the buckets that are not a multiple of the common factor will hash much less key than those are.

3. The *Nearest neighbor* mentioned in the class is an example of a greedy algorithm which is locally optimal while not being globally optimal. And the discussion in the class provide all the necessary details to support my claim.

References

[1] Proof of Greedy Algorithm in Knapsack Problem, https://www.cnblogs.com/hapjin/p/5575112.html