COMP5711 Assignment 3

WONG Yuk Chun (20419764) ycwongal@connect.ust.hk

MR 8.22

For fixed x and y, when collision occur, then

$$\begin{aligned} h(x) &= h(y) \\ ax \bmod p &\equiv ay \bmod p \pmod{n} \end{aligned}$$

Let $s = ax \bmod p$ and $t = ay \bmod p$, since $x \neq y$ we have

$$\begin{aligned} &\begin{cases} s - t & \equiv a(x-y) \pmod{p} \\ s & \equiv t \pmod{n} \end{cases} \\ \implies &\begin{cases} a & \equiv (x-y)^{-1}(s-t) \pmod{p} \\ s & \equiv t \pmod{n} \end{cases} \end{aligned}$$

Since p is prime, $x \neq y$, then $a \in [0, p-1]$ that $h_a(x) = s$ and $h_a(y) = t$ is unique. Note that $s \neq t$ or else $a = 0$ and h_0 maps all parameters to 0 which is obviously not a good hash function. Since $|H|$ is equivalent with the number of choice of a, $|H|$ is equivalent with number of choice of $(s-t)$ such that $(s-t) \bmod p \in [p-1]$ and $s \neq t$, then $|H| = p - 1$. To calculate number of h such that $h(x) = h(y)$, count number of $(s-t)$ such that $(s-t) \in [1, p-1]$ and $s \equiv t \pmod{n}$, which is $\leq 2(p-1)/n$.

Then the required probability

$$P_{h\in H}(h(x) = h(y)) \leq \frac{2(p-1)/n}{p-1} = \frac{2}{n}$$

KT 13.14

The algorithm is, for each job: assign its $2n$ processes to the two machines with same probability independently, until the assignment is nearly balanced. Consider a job, define X be the number of processes assigned to M_1, we have

$$X = \sum_{i=1}^{2n} X_i, \text{ where } X_i = \begin{cases} 1 & \text{if process } i \text{ is assigned to } M_1 \\ 0 & \text{if process } i \text{ is assigned to } M_2 \end{cases}$$

and

$$\mu = E[X] = n$$

Then by Chernoff inequality, the probability of a assignment of this job is not nearly balance is

$$\begin{aligned} P(\text{not nearly balanced}) &= P(X > \frac{4}{3}n) + P(X < \frac{2}{3}n) \\ &\leq P(X \geq (1 + \frac{1}{3})\mu) + P(X \leq (1 - \frac{1}{3})\mu) \\ &\leq \exp(-\mu(1/3)^2/3) + \exp(-\mu(1/3)^2/2) \\ &\leq 2 \times 0.97^n \end{aligned}$$

By union bound, the probability that at least one of jobs fails the nearly balanced condition is at most $2n \times 0.97^n$, which is almost zero for big enough n. Then we can expect one assignment asymptotically to make one job nearly balanced, hence the algorithm can run in linear time.

KT 13.15

Let N be the sample size, a and b be the $(\frac{1}{2} - \epsilon)n$ th number in sorted ascending and descending order respectively. To have the median of sample satisfy the condition, we need to have less than $N/2$ samples less than a and less than $N/2$ samples greater than b. Consider the first condition, let

$$X = \sum X_i \text{ ,where } X_i = \begin{cases} 1 & x_i < a,\, x_i \text{ is the i-th sample} \\ 0 & \text{otherwise} \end{cases}$$

$$E[X] = (\frac{1}{2} - \epsilon)N$$

Then apply Chernoff inequality, assume $N/2 \geq (1+\epsilon)E[X] \implies \epsilon \leq \frac{1}{2}$, the probability to not satisfy the first condition is given by

$$P(X > N/2) \leq P(X > (1+\epsilon)(\frac{1}{2} - \epsilon)N) \leq \exp(-(\frac{1}{2} - \epsilon)N\epsilon^2 \frac{1}{3}) \leq \frac{\delta}{2}$$

$$\implies N \geq \frac{6 \ln \frac{2}{\delta}}{\epsilon^2}$$

Similarly, we have the same result for the second condition.

By union bound, the sum of probability to fail first and second condition is no greater than $\delta/2 + \delta/2 = \delta$. Hence the sample size is $\Omega(\frac{1}{\epsilon^2} \ln \frac{2}{\delta})$.

If use pairwise independent hash function , more samples is required since Chernoff inequality requires all random variables to be independent.

RIC

Define Y as the sorted list and X be the set of x. Initialize $Y = [y_0]$ where all x shall be inserted after y_0, y_0 is a dummy head, and initialize pointers to and from y_0 and x, which the process takes linear time.

At the middle of insertion process, which X and Y is non empty, suppose we are inserting $y \in X$ to Y behind y_i. Define $X_i \subset X$, X_i is the set of x that is pointed by y_i and shall be inserted after y_i. We relabel the pointers associated with X_i. For instance, for each $x \in X_i$, if $x < y$, keep the original pointers of x points to y_i and y_i points to x, else update the pointers so that x points to y and y points to x. The time for each incremental step is linear to the size of X_i, and size of X_i will be shrink by at least 1.

The proof is similar to randomized quick sort.

For each insertion we divide a partition in X to 2 smaller partitions. In later steps we further insert elements in the two partitions using the same policy, so the process have a recurrence relationship. Define a good division as the choice of newly inserted element can divide the partition into 2 smaller ones of 25%~75% of original, then we have the recurrence

$$T(n) = T(\frac{1}{4}n) + T(\frac{3}{4}n) + n = \Theta(n \log n)$$

Since the element is chosen uniformly, we have probability of 0.5 to have a good division, then we can expect a good division after 2 divisions, so that the largest partition will shrink by at least 1/4. Hence the expected run time of both good and bad division are both $O(n \log n)$

> I am not sure whether is it appropriate to use proof above, my friend taught me to use backward analysis. When j elements have inserted to Y, there are $n-j$ elements left in X. To step one step back, remove one element in Y, let's say y, it is equal likely to be any item in Y so each have $1/j$ probability, and only those $x \in X$ which is just greater than y (y is the greatest element in Y less than X) need to update its pointer, where each update is $O(1)$. So each backward step has expected run time of $O((n-j)/j)$. To sum up, the total expected cost is $O(\sum_{j=1}^{n}(n-j)/j) = O(n \log n)$