COMP 5711 - Advanced Algorithms Written Assignment # 3

ZHAO, Xi
12256508
xzhaoca@connect.ust.hk

November 19, 2023

MR8.22

Let x and$x$ be two distinct elements of U. $|H| = p-1$, we count the number of a, N_a, such that $h_a(x) = h_a(y)$, then show that $N_a/(p-1) \leq 2/n$. Assume $h_a(x) = t$ and $ax = u \mod p$, for a $y \neq x$ in $[u]$, let $ay = v \mod p$. Since $u = v = t \mod n$, for each value of t, $v \in [p]$ can take at most $\lceil p/n \rceil \leq p/n$ different values. So, $N_a/(p-1) \leq (p/n)/(p-1) \leq 2/n$.

RIC

In this list structure, each node x may have multiple successor nodes y (that is, y are inserted after x) such that $x < y$ and there is only a head node (the smallest element) when the list is not empty. To insert a new element a, we find the node t such that a should be inserted after t. We first consider the head node h in the list as t and compare whether $a < t$. If so, a should be the new head node and we maintain the pointers between a and h; otherwise, we compare a and all of t's successors ys and a. If no y in t's successors is less than a, we find the required t and insert a after t; otherwise, there exists a y that $y < o$, we consider the largest such y as the next t and repeat this procedure.

Running Time: We denote the layer of a node a as the length of path from the head node h to a. As the elements are inserted in a random order, the expected layer of the i-th largest points in the array is $O(\log i)$ and each node has expected $O(1)$ successor. So, the expected time of inserting an element is $O(\log n)$ and all the running time is $O(n \log n)$.

KT13.14

We independently assign each process i a label L_i as 0 or 1 randomly. For a job J, we assign each process i in J's $2n$ processes to machine M_1 if L_i is 0 and to machine M_2 if L_i is 1. Denote $L = \sum_{t \in J} L_t$ and $\mathrm{E}\, L = n$, the probability that the assignment in J is not nearly balanced is $p = \Pr[L > 4n/3] + \Pr[L < 2n/3] \leq 2\exp(-n/3)$ according to the Chernoff inequality. So, the probability that any of n jobs is not nearly balanced is at most $p_t = np \leq 2n\exp(-n/3)$. We can find an enough large n to make $P - t \leq 0.5$ since $f(n) = 2n\exp(-n/3)$ decreases with n and $f(+\infty) = 0$.

After such an assignment, we check whether all jobs are nearly balanced, which takes $O(n^2)$ time. If not, we repeat such a process. The expected repeated number is $1/p_t = 2$. So, the expected running time is still polynomial with n.

KT13.15

We imagine dividing the set S into $2L+1$ quantiles $Q_i, Q_2, \cdots, Q_{2L+1}$, and each quantile contains at least $n_i = \lfloor n/(2L+1) \rfloor$ points. We randomly choose $(2L+1)b$ points from S as sampling set S'. So, the expected number of points falling in a quantile is $\mathrm{E}\, X = b$. Given a small positive number t,

$$p = \Pr[|X - \mathrm{E}\, X| \geq t\, \mathrm{E}\, X] \leq 2\exp(-bt^2/3)$$

according to the Chernoff inequality. If such a condition satisfies in all $2L$ quantiles expect for the median quantiles Q_{L+1}, the first L quantiles will have at most $(1+t)Lb$ points in S' and the last L quantiles will have at most $(1+t)Lb$ points in S' too. By setting b such that

$$Lb < (1+t)Lb \leq (1+L)b,$$

the median of S' will belong to Q_{L+1}. In this case, $t \leq 1/L$. Then, to make sure any point in Q_{L+1} is a ϵ-approximate median, we should ensure $(L+1/2)n_i - Ln_i \leq \epsilon n$, indicating

$$n_i \leq 2\epsilon n.$$

So, $n/(2L+1) \leq 2\epsilon n$ and $L = \lceil \frac{1}{4\epsilon} \rceil$ is fine.

Finally, the probability that each of $2L$ quantiles expect for the median quantiles Q_{L+1} has more than $(1+t)Lb$ points in S' is $p_t \leq 2Lp = 4L\exp(-b/(3L^2))$. To ensure $p_t \leq \delta$, we have $b \geq 3L^2 \ln(4L/\delta)$. Therefore, the required number of points to be sampled is

$$|S'| = (2L+1)b = 3L^2(2L+1)\ln(4L/\delta),$$

which is $O((1/\epsilon)^3(\ln(1/\epsilon) + \ln(1/\delta)))$.

When there is a pairwise independent hash function with $h_{a,b} = ax + b \mod p$, the close points are often mapped into different hash buckets. So, we can choose a hash bucket which contains the most number of points and use the points in it as S'.