COMP5711 Assignment 4

> WONG Yuk Chun 20419764 ycwongal@connect.ust.hk

SKI

a. Deterministic strategy

One strategy is rent until day n.

Suppose go for less than or equals to n day, obviously we reach the optimal result that is competitive ration is 1.

If we buy before $n+1$ day, we spend $2n$ dollars, and the optimal is buy at the very first day, by which we spend n dollars only, hence the competitive is at most 2.

b. Adversarial argument

Suppose we buy the ski on t^{th} day and immediately stop skiing as an adversarial case. If $t \leq n$ obviously the best case is rent for all t days which costs t dollars, otherwise it is the best to buy on the very first day which cost n dollars. Then the competitive ratio $r(t)$ will be

$$r(t) = \begin{cases} \frac{n+(t-1)}{t} & t \leq n \\ \frac{n+(t-1)}{n} & t \geq n \end{cases}$$

By solving this equation, we have $t = n$ to minimize the competitive ratio, where $r(n) = 2 - 1/n$, then there is no deterministic strategy to have better worse case competitive ratio than this.

c. Randomized strategy

For first $\frac{n}{2}$ days, we rent. Between $\frac{n}{2}$ and n-th day, we buy with a probability of $\frac{2}{n}$, then we buy the ski. Since $\sum_{\frac{n}{2} \leq t < n} \frac{2}{n} = 1$, we must had bought the ski on n-th day. Therefore, the expected cost by the algorithm is

$$\begin{aligned} c &= \frac{n}{2} + \sum_{\frac{n}{2} \leq t < n} [(t+n)\frac{2}{n}] \\ &= \frac{n}{2} + [\frac{n+\frac{n}{2}}{2} + (n - \frac{n}{2})]\frac{2}{n} \\ &= \frac{n}{2} + [\frac{3}{4} + \frac{n^2}{2}]\frac{2}{n} \\ &= \frac{3}{2}(n + \frac{1}{n}) \end{aligned}$$

Hence the expected competitive ratio is $\frac{3}{2}(1 + \frac{1}{n^2})$ which is better than the deterministic case.

MG

a.

Since there are M elements left in the counter, in the stream there are only $N-M$ elements can be deleted, each decrement event corresponds to $k+1$ deletion of distinct elements in the stream, so there can be at most $\frac{N-M}{k+1}$ decrements on the key. Therefore the error is upper bounded by $\frac{N-M}{k+1}$.

b.

Since count of each element is differed by at most $\frac{N-M}{k+1}$, sum the count of first t most frequent element will be less then all count plus t maximum error.

$$\sum_{i=1}^{t} f_i \leq M + \frac{N-M}{k+1}t$$

then

$$\begin{aligned} N^{res(t)} &= N - \sum_{i=1}^{t} f_i \\ &\geq N - M - \frac{N-M}{k+1}t \\ &= (N-M)(1 - \frac{1}{k+1}t) \\ &= (N-M)(k+1-t)\frac{1}{k+1} \\ \implies error \leq \frac{N-M}{k+1} &\leq \frac{N^{res(t)}}{k+1-t} \end{aligned}$$

Parallel

The idea is, first map all elements that equal to 1 to its index, otherwise to infinity, then find the smallest value in the array (aka the first index of 1).

1. In parallel, set $B[i] = \begin{cases} i & A[i] = 1 \\ \infty & \text{otherwise} \end{cases}$, set $N =$ length of A
2. If $N = 1$, then $B[0]$ is the first index
3. In parallel, for i from 0 to half N, set $C[i] = \min(B[2i], B[2i+1])$. Treat $B[2i+1]$ as ∞ if it is not defined (aka case N odd)
4. Set $N =$ half N, $B = C$
5. Repeat 2, 3, 4, 5

Each step 1,2,3,4 is $O(1)$ in parallel, and 2,3,4,5 repeat for $O(\log n)$ times. Hence total parallel run time is $O(\log n)$

In sequential, step 1 takes $O(n)$ times. Total run time of 2,3,4 = $\sum(O(1) + O(N) + O(N)) = \sum O(N)$, but each time N is dropped by half so the run time is $O(n) + O(n/2) + O(n/4) + \ldots + O(1) = O(2n) = O(n)$. So the total work is $O(n)$.