PAGE
DATE : / /

PROBABILISTIC ANALYSIS AND RANDOMIZED ALGOS.

5.1 Hiring problem

- want to hire assistant.
- employment agency sends n people randomly
- if current person > current assistant, fire current assistant and hire new.
- Find cost of this problem

C_i = interview cost

C_h = hiring cost.

Cost = $nC_i + mC_h$

m = no. of hired.

Worst case : Increasing order of rank.

Cost = $nC_i + nC_h$ at $m = n$.

Probabilistic analysis.

We assume that there is equal probabity of each permutation of occurrence of inputs.

In probabilistic analysis we average over all possible inputs so we call average case running time.

We call the probabity of permutation to occour equally as uniform random permutation.

Randomized algorithms.

Suppose the agency sends people in increasing order to make profit. To not allow this to happen we use algorithms to permutate the array.

2 ways :

1. Give random weight to person and sort accordingly.
2. Swap randomly.

5.2 Indicator random variable.

Indication random valiable associated with event A is

$$I\{A\} = \begin{cases} 1 & \text{if } A \text{ occours} \\ 0 & \text{if } A \text{ doesnt occurs.} \end{cases}$$

eg.

X_H = indicator RV coin coming up heads.

$$X_H = I\{H\}$$
$$= \begin{cases} 1 & \text{if Head come} \\ 0 & \text{if not come.} \end{cases}$$

Expected no. of heads.

$$E[X_H] = E[I\{H\}]$$
$$= 1 \cdot Pr\{H\} + 0 \cdot Pr\{T\}$$
$$= 1 \cdot 1/2 + 0 \cdot 1/2)$$
$$= 1/2.$$

Lemma

Given sample space S and event A let $X_A = I\{A\}$

Then $E[X_A] = Pr\{A\}$.

Hiring problem analysis.

$$x_i = \begin{cases} 1 & \text{if candidate } i \text{ is hired} \\ 0 & \text{if candidat } i \text{ is not hired} \end{cases}$$

X = IRV whose value = no. of time we hire

$$X = X_1 + X_2 + \ldots X_n$$

$$E[X] = E\left[\sum_{i=1}^{n} x_i\right]$$

$$= \sum_{i=1}^{n} E[x_i]$$

$$= \sum_{i=1}^{n} 1/i = \ln n + O(1)$$

since for i^{th} person prob. it selected is $1/i$.

5.3 Randomized algorithm

This should make sure that each possible permutation can be formed.

5.4 The birthday paradox problem.

How many people should be there in room for 50% chance of 2 people having same b'day.

Let there be k people and $n = 365$ days.

So we assume that b'days are uniformly distributed

$$Pr(b_i = r) = \frac{1}{n} \quad i = 1 \ldots k, \quad r = 1, 2, 3 \ldots n$$

$$P(b_i = r \text{ and } b_j = 1) = Pr(b_i = 1)\, Pr(b_j = r)$$

$$= 1/n^2.$$

$$Pr\{b_i = b_j\} = \sum_{r=1}^{n} Pr\{b_i = r, b_j = r\}$$

$$= \sum_{r=1}^{n} (1/n^2) = \frac{1}{n}.$$

Now consider the case of atleast 2, have same b'day.

$$Pr\{B_k\} = Pr\{B_{k-1}\} Pr\{A_k \mid B_{k-1}\}$$

$$= Pr\{B_{k-2}\} Pr\{A_{k-1} \mid B_{k-2}\} Pr\{A_k \mid B_{k-1}\}.$$

$$= 1 \left(\frac{n-1}{n}\right)\left(\frac{n-2}{n}\right) \cdots \left(\frac{n-k+1}{n}\right)$$

$$= 1 \cdot \left(1 - \frac{1}{n}\right)\left(1 - \frac{2}{n}\right) \cdots \left(1 - \frac{k-1}{n}\right)$$

as $1 + x < e^x$.

$$Pr\{B_k\} \leq e^{-1/n} e^{-2/n} \cdots e^{-(k-1)/n}.$$

$$\leq e^{-\sum_{i=1}^{k-1} i/n}$$

$$\leq e^{-k(k-1)/2n} \leq 1/2$$

Now for 50%

$$k(k-1) \geq 2n \ln 2.$$

$$k \geq (1 + \sqrt{1 + 8(\ln 2) n})/2$$

for $n = 365$

$$k \geq 23.$$

IRV

For each pair i, j of k people

$$X_{ij} = \begin{cases} 1 & \text{if } i, j \text{ have same b'day} \\ 0 & \text{otherwise} \end{cases}$$

$$E[X_{ij}] = Pr\{ i, j \text{ have same b'day} \}$$
$$= 1/n$$

$$X = \sum_{i=1}^{k} \sum_{j=i+1}^{k} X_{ij}$$

$$= \sum_{i=1}^{k} \sum_{j=i+1}^{k} X_{ij}.$$

$$= \binom{k}{2} \frac{1}{2}$$

$$= \frac{k(k-1)}{2n}.$$

$k(k-1) \geq 2n$

if k is atleast $\sqrt{2n} + 1$

for this $k = 28$.

$O(\sqrt{n})$

Amortized analysis.

In an amortized analysis, the time required to perform a sequence of data structure operation is average over all operation.

Can be used to show that average cost of an operation is small, even though a single operation within the sequence might be expensive.

Amortized vs average case.

In amortized analysis probability is not there but in average case analysis probability is there.

Techniques:

a) Aggregate method.

In this for all n, a sequence of n operations taken worst-case time T(n) in total. In the worst case, the average cost, or amortized cost, per operation is T(n)/n.

Eg. Push pop multipop

Push → O(1)

Pop → O(1)

multipop → max(O(n,1))

Now if there are n push/pop/multipop operation

worst case = $O(n^2)$ for multipop.

But amortized:

consider if stack contains n elements the at max there can be n push or n pop, and a multipop can atmost pop n→

So it is $O(n)$. Now consider it has n operation so $\frac{O(n)}{n} \longrightarrow O(1)$.

Increment binary problem.

Algo :

```
Increment (A, k) {
  int i=0;
  while ( i<k and A[i]==1)
      A[i]=0
      i+=1;
  } else if (i<k)
      A[i] = 1
}
```

$O(k)$

If increment n times

worst case = $O(nk)$

But for n increment the i^{th} bit changes $\left[\frac{n}{2^i}\right]$

time. i.e for $n=4$

$A[0]$: 4 times

$A[1]$: 2 times.

:

for $i=0 \ldots \lfloor \log n \rfloor$ $A[i]$ flips $\lceil n/2^i \rceil$ times.

Actual time

$$\sum_{i=0}^{\lfloor \log n \rfloor} \left\lceil \frac{n}{2^i} \right\rceil \leq n \sum_{i=0}^{\infty} 1/2^i = n \; 1/1-1/2 = 2n$$

$= O(n)$

So each operation $O(1)$.

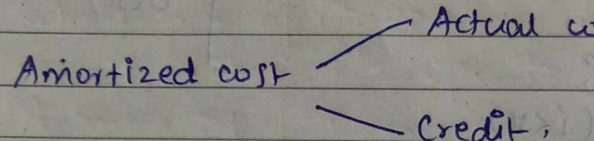
PAGE
DATE : / /

The accounting method.

In this method, we assign differing charges to diff. operations, with some operation charged more or less than they actually cost. The amount we charge an operation is called amortized cost.

Credit = (An op. amortized cost) − (Actual cost).

It can be use for operation whose amortized cost > actual cost.



In aggregate analysis all method has same amortized cost.

In accounting method they have diff. amortized cost.

If $\hat{c_i}$ = amortized cost.

c_i = Actual cost

Total credit = $\sum_{i=1}^{n} \hat{c_i} - \sum_{i=1}^{n} c_i$.

Eg. Stack operation.

push( )	O(1)
pop ( )	O(1)
multipop	O(min (n,k))

Suppose we assign amortized cost as follows.

push (x) − 2 1 (actual) + 1 (credit)

when we pop we use this credit.

pop ( ) − 0

multipop ( ) − 0

Thus for any sequence of n push, pop, and multipop operation, the total amortized cost is a n upper bound on the total actual cost. Since total AC is O(n) which is also the actual cost.

Binary counter.

```
Increment(A,n) {
    i=0
    while ( i<n and A[i]==1) {
        A[i] = 0          ]
        i+=1;             ] - 0 cost of reseting.
    }
    if i<n
        A[i]=1  — cost of setting bit = 2
                  i.e. 1 + 1 credit for
                       reset.
}
```

Thus for n increment amortized cost is O(n)

The potential method :-

This method represents the prepaid work as 'potential energy' that can be released to pay for future operation.

The potential is associated with the data structure as a whole rather, than with specific objects within the data structure

D_0 = Initial DS on which n op. are performed.
c_i = Actual cost of i^{th} op.
D_i = The DS after i^{th} op. to D_{i-1}.
ϕ : A potential fn.
$\phi(D_i)$ is potential of D_i.

$\hat{c}_i$: Amortized cost of ith operation

$$\hat{c}_i = c_i + \phi(D_i) - \phi(D_{i-1}).$$

Actual + Potential diff

Total amortized.

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \phi(D_i) - \phi(D_{i-1}))$$

$$= \sum_{i=1}^{n} c_i + \sum_{i=1}^{n} (\phi(D_i) - \phi(D_{i-1}))$$

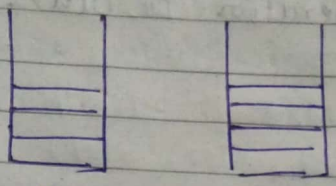
$$= \sum_{i=1}^{n} c_i + \phi(D_n) - \phi(D_0)$$

if $\phi(D_n) \geq \phi(D_0)$.

Now we might not know how many op. are there so we keep $\phi(D_i) \geq \phi(D_{i-1})$ so that $\phi(D_n) \geq \phi(D_0)$.

PAGE
DATE: / /

Eg. Stack operations.



Push.

$= \phi(D_i) - \phi(D_{i-1})$

$= (S+1) - S.$

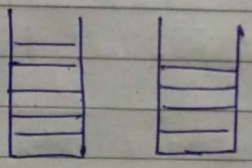
$= 4 - 3$

$= 1.$

So amortized cost of push $= C_i + 1$

$= 2.$

Pop



$\phi(D_{i-1})$ $\phi(D_i$

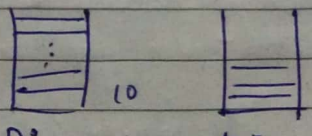
$\phi(D_i) - \phi(D_i)$

$= -1$

Amortized cost $= 1 + -1$

$= 0.$

multipop



D_{i-1} D_i

$\hat{C}_i = C_i + \phi(D_i) - \phi(D_{i-1})$

$= k - k'$

$= 0.$

PAGE
DATE : / /

So for a push or and pop operation worst case on an operation is O(n).

Fast fourier transform.

Polynomial addition take $O(n)$ but multiplication take $O(n^2)$.

If

$x(n) = \{1, 2, 3, 4\}$ $N=4$

$$x(k) = \sum_{n=0}^{3} x(n) W_N^{nk}.$$

$$W_N = e^{-j2\pi/n}.$$

$$x(k) = x(0)W_4^0 + x(1)W_4^k + x(2)W_4^{2k} + x(3)W_4^{3k}.$$

$x(0)$

$x(1)$

$x(2)$

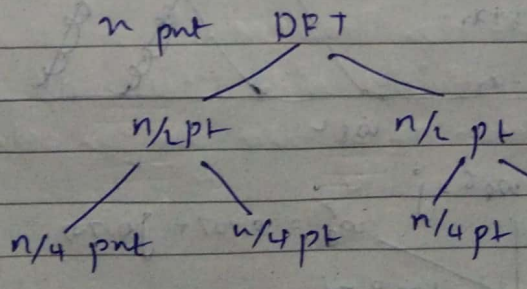
$x(3)$

for this complex multiplication = $4 \times 4 = 16$

complex addition = $3 \times 4 = 12$.

If $n = 256$ then CM $= 256 \times 256$.

Cooley and Tukey devised a divide and conquer to find



So $n = 2^x$ (multipe of 2)

8 pt DIT-FFT flow graph.

8 pt $x(n) = \{x(0), x(1), x(2), x(3), x(4), x(5), x(6), x(7)\}$

Divide

4 pt $x_e(n) = \{x(0), x(2), x(4), x(6)\}$ $N = 4$

4 pt $x_o(n) = \{x(1), x(3), x(5), x(7)\}$ $N = 4$

2 pt $x_1(n) = \{x(0), x(4)\}$ $x_2(n) = \{x(2), x(6)\}$

2 pt $x_3(n) = \{x(1), x(5)\}$ $x_4(n) = \{x(3), x(7)\}$