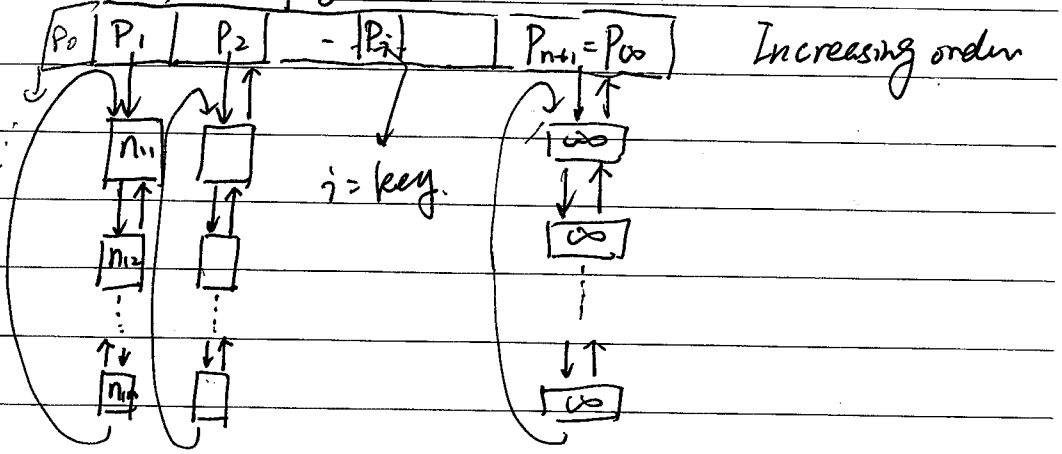
HW5 Yue Li

No. 1

Date 2008/11/7

1. Ex 24.3.6.

Define an array of pointers $P = [P_0, P_1, P_2, \ldots P_n, P_{n+1}]$ where each pointer P_i is the head pointer which points to a linked list, see fig below



Each linked list is a doubley linked, circular list, and the linked list that P_i points to contains all the nodes x in the graph whose $d[x] = i$.

Because $E \rightarrow \{0, 1, \ldots, W\}$, Therefore the largest possible $d[x]$ will be $W(|V|-1)$. Thus to build the pointer array, we need $W(|V|-1)$ pointers, plus the last one P_∞ which denotes the linked list containing nodes whose $d = \infty$. Therefore, under such priorityqueue, INSERT can be done in $O(1)$ time, DECREASE-KEY can be done in $O(1)$ time, $\therefore$ the INITIALIZE-SINGLE-SOURCE costs $O(V)$ times, and the total RELAX operations cost $O(E)$ times.

We now look at Extract-Min, consider a sequence of $|V|$ Extract-Min after performing $|V|$ inserts as the DIJKSTRA does, since there are at most $|V|$ insertions,

No.

Date

(pointer list) ?

and the list is key for each pointer is increasing, therefore we use min pointer to keep track of the location of index after each Extract-Min, therefore, to run Extract-Min at second time can just start from the min pointer points to. need to clarify the above. Index the

Therefore, the sequence of $|V|$ EXTRACT-MIN costs $O(W|+W(|V|-1))$ time.

∴ The total time for the modified Dijkstra algo becomes:

$O(|V|+|E|+|V|+W|V|-W)$

$=O(WV+E)$

2. Problem 24-1. Yen's improvement to Bellman-Ford.

(a) Proof. (A) According to the definition of E_f:

$E_f = \{(v_i, v_j) \in E : i < j\}$, edges are ordered from small index to large one, therefor G_f is topologically sorted,

(B) Assume G_f is cyclic, thus there's a ~~circle~~ cycle in G_f, which means there must be some (v_i, v_j) where $i > j$, this contradicts ~~with~~ the fact that $i < j$.

∴ G_f is acyclic.

From (1) and (2), G_f is acyclic and topologically sorted.

Same proof with G_b.

(c) From part (b), we know only $\lceil |V|/2 \rceil$ passes over edges. therefore, the new running time becomes

$$O(\frac{V}{2} \cdot E) = O(VE)$$

∴ Equals the previous running time.

3. Problem 25-1. Transitive closure of a dynamic graph.

(a). ∵ Transitive closure can be represented by boolean matrix, therefore, let

$$TC[i,j] = \begin{cases} 1 & \text{if there is a path for } i \text{ to } j \\ 0 & \text{otherwise} \end{cases}$$

$1 \le i, j \le |V|$

The update algo: (assume we add in edge (u, v),

1. for $i = 1, 2, \ldots$ to $|V|$
2. for $j = 1, 2, \ldots$ to $|V|$
   if $TC[i,u] = TC[v,j] = 1$
   then $TC[i,j] = 1$

The running time is thus $O(|V|^2)$

7.

Ex 5.2-4.

Let $X_i = \begin{cases} 1 & \text{if the } i\text{th customer gets back his hat} \\ 0 & \text{otherwise} \end{cases}$

$\therefore P(X_i = 1) = 1/n$ ← Why?

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

$= \sum_{i=1}^{n} 1/n$

$= 1$

Because the hat-check person gives the hats back to the customers in a random order. There are $n!$ permutations for the return order. $(n-1)!$ of them have i^{th} entry equal to i. $\frac{(n-1)!}{n!} = \frac{1}{n}$.

$\therefore$ 1 customer is expected to get his hat back.

8. Ex 7.4-4 According to Page 158,

$E[X] = \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \frac{2}{j-i+1}$

$= \sum_{i=1}^{n-1} \sum_{k=1}^{n-i} \frac{2}{k+1} > \sum_{i=1}^{n-1} \sum_{k=1}^{n-1} \frac{1}{k+1} = \sum_{i=1}^{n-1} \lg n$.

$\therefore E[X]$ is $\Omega(n \lg n)$.

Delijia®

Date

9. Ex 7.4-5.

① The Algorithm starts by running quick sort, it keeps on running until the number of each subproblem becomes k, then apply insertion sort. the elements in

Therefore, the number of "final" subproblems is n/k

solving each subproblem with insertion sort costs $O(k^2)$

$\therefore$ Totally $n/k \times O(k^2) = O(nk)$

? → divide progress.

For the quicksort (divide progress):

the height of the tree is $\lg n - \lg k = \lg \frac{n}{k}$

For each level of the tree, the total cost is $O(n)$

$\therefore$ Totally $\lg \frac{n}{k} \cdot O(n) = O(n \cdot \lg \frac{n}{k})$

$\therefore$ Running time is $O(nk + n \cdot \lg \frac{n}{k})$.

② Let $f(k) = nk + n \cdot \lg \frac{n}{k}$, to choose k for optimization, we calculate $f'(k) = 0$ explain

$k = 1/\ln 2$. (Theoretically)

Practically, depends on the array's property.

for example: if the array is almost sorted,

k should be as large as possible.

To compute maximal value of function $f(k)$, one approach is to let the differential $f'(k) = 0$, solve the constraint obtaining k, which makes $f(k)$ maximal.

Ex. C.3-3 Bet \$1 on any number 1 to 6. Roll 3 dice.

If chosen number doesn't appear, lose \$1.

If " " appears once, win \$1.

If " " " twice, win \$2.

If " " " three times, win \$3.

What is expected gain? Fix chosen number to be i.

Let X = amount won (or lost).

$$E[X] = \sum_{v=-1,1,2,3} v \cdot \Pr[X=v]$$

Need to calculate $\Pr[X=-1], \ldots, \Pr[X=3]$.

$$\Pr[X=-1] = \Pr[i \text{ doesn't appear}] = \frac{5}{6}\cdot\frac{5}{6}\cdot\frac{5}{6} = \frac{125}{216}$$

$$\Pr[X=1] = \Pr[\text{exactly 1 die rolls } i] = \frac{1}{6}\cdot\frac{5}{6}\cdot\frac{5}{6}\cdot 3 = \frac{75}{216}$$

(3 = # of choices for the die that rolls i)

$$\Pr[X=2] = \Pr[\text{exactly 1 die does } \underline{\text{not}} \text{ roll } i] = \frac{1}{6}\cdot\frac{1}{6}\cdot\frac{5}{6}\cdot 3 = \frac{15}{216}$$

$$\Pr[X=3] = \Pr[\text{all 3 dice roll } i] = \frac{1}{6}\cdot\frac{1}{6}\cdot\frac{1}{6} = \frac{1}{216}$$

So $$E[X] = (-1)\cdot\frac{125}{216} + 1\cdot\frac{75}{216} + 2\cdot\frac{15}{216} + 3\cdot\frac{1}{216}$$

$$= \frac{-19}{216}$$

(expect to lose money)

Ex. C-2-3: Deck of 10 cards, numbered 1 to 10, is shuffled.

Three cards are removed one at a time. What is the probability that the 3 cards are selected in increasing order?

$$\Pr(1^{st} < 2^{nd} \text{ and } 2^{nd} < 3^{rd})$$

$$= \sum_{\substack{\text{all subsets} \\ \{a,b,c\} \text{ of} \\ \text{the cards}}} \Pr(\{a,b,c\} \text{ is chosen}) \cdot \Pr(\text{in sorted order} \mid \{a,b,c\} \text{chosen})$$

$$= \sum_{\substack{\text{all subsets} \\ \{a,b,c\} \text{ of} \\ \text{the cards}}} \frac{1}{\binom{10}{3}} \cdot \frac{1}{3!}$$

There are 3! different permutations of $\{a,b,c\}$, all are equally likely, and only one is in sorted order

$$= \binom{10}{3} \cdot \frac{1}{\binom{10}{3}} \cdot \frac{1}{3!}$$

$$= \frac{1}{3!} = \boxed{\frac{1}{6}}.$$

Ex C.3-2 $A[1..n]$ of n distinct numbers is randomly ordered, with each permutation equally likely. What is expectation of index of maximum element in array?

Let X = index of max. index.

$$E[X] = \sum_{v=1}^{n} v \cdot \Pr[X = v]$$

$\Pr[X = v] = \frac{1}{n}$, since it is equally likely that the max is anywhere

$$E[X] = \sum_{v=1}^{n} v \cdot \frac{1}{n}$$

$$= \frac{1}{n} \cdot \sum_{v=1}^{n} v = \frac{1}{n} \cdot \frac{n(n+1)}{2}$$

$$= \boxed{\frac{n+1}{2}}$$

min is the same.