Hiring Problem and Generating Random Permutations

Andreas Klappenecker

Partially based on slides by Prof. Welch

- You need to hire a new employee.
- The headhunter sends you a different applicant every day for n days.
- If the applicant is better than the current employee then fire the current employee and hire the applicant.
- Firing and hiring is expensive.
- How expensive is the whole process?

- Worst case is when the headhunter sends you the n applicants in increasing order of goodness.
- Then you hire (and fire) each one in turn: n hires.

- Best case is when the headhunter sends you the best applicant on the first day.
- Total cost is just 1 (fire and hire once).

- What about the average cost?
- An input to the hiring problem is an ordering of the n applicants.
- There are n! different inputs.
- Assume there is some distribution on the inputs
  - for instance, each ordering is equally likely
  - but other distributions are also possible
- Average cost is expected value…

- We want to know the expected cost of our hiring algorithm, in terms of how many times we hire an applicant
- Elementary event s is a sequence of the n applicants
- Sample space is all n! sequences of applicants
- Assume uniform distribution, so each sequence is equally likely, i.e., has probability 1/n!
- Random variable X(s) is the number of applicants that are hired, given the input sequence s
- What is E[X]?

- Break the problem down using indicator random variables and properties of expectation

- Change viewpoint: instead of one random variable that counts how many applicants are hired, consider n random variables, each one keeping track of whether or not a particular applicant is hired.

- Indicator random variable X_i for applicant i:
  1 if applicant i is hired, 0 otherwise

- Important fact: $X = X_1 + X_2 + \ldots + X_n$
  - number hired is sum of all the indicator r.v.'s
- Important fact:
  - $E[X_i]$ = Pr["applicant i is hired"]
  - Why? Plug in definition of expected value.
- Probability of hiring i is probability that i is better than the previous i-1 applicants…

- Suppose n = 4 and i = 3.
- In what fraction of all the inputs is the 3rd applicant better than the 2 previous ones?

1234	2134	3124	4123
1243	2143	3142	4132
1324	2314	3214	4213
1342	2341	3241	4231
1423	2413	3412	4312
1432	2431	3421	4321

- Suppose n = 4 and i = 3.
- In what fraction of all the inputs is the 3rd applicant better than the 2 previous ones?

1234	2134	3124	4123
1243	2143	3142	4132
1324	2314	3214	4213
1342	2341	3241	4231
1423	2413	3412	4312
1432	2431	3421	4321

8/24 = 1/3

- In general, since all permutations are equally likely, if we only consider the first i applicants, the largest of them is equally likely to occur in each of the i positions.

- Thus $Pr[X_i = 1] = 1/i$.

- Recall that X is random variable equal to the number of hires

- Recall that X is random variable equal to the number of hires
- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the i-th applicant is hired)

- Recall that X is random variable equal to the number of hires
- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the i-th applicant is hired)
- $E[X] = E[\sum X_i]$

- Recall that X is random variable equal to the number of hires

- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the i-th applicant is hired)

- $E[X] = E[\sum X_i]$

  $= \sum E[X_i]$, by property of E

- Recall that X is random variable equal to the number of hires

- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the i-th applicant is hired)

- $E[X] = E[\sum X_i]$

  $= \sum E[X_i]$, by property of E

  $= \sum Pr[X_i = 1]$, by property of X_i

- Recall that X is random variable equal to the number of hires

- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the i-th applicant is hired)

- $E[X] = E[\sum X_i]$

  $= \sum E[X_i]$, by property of E

  $= \sum Pr[X_i = 1]$, by property of X_i

  $= \sum 1/i$, by argument on previous slide

- Recall that X is random variable equal to the number of hires
- Recall that X = the sum of the X_i's (each X_i is the random variable that tells whether or not the i-th applicant is hired)
- $E[X] = E[\sum X_i]$

$= \sum E[X_i]$, by property of E

$= \sum Pr[X_i = 1]$, by property of X_i

$= \sum 1/i$, by argument on previous slide

$\leq \ln n + 1$, by formula for harmonic number

- So average number of hires is ln n, which is much better than worst case number (n).
- But this relies on the headhunter sending you the applicants in random order.
- What if you cannot rely on that?
  - maybe headhunter always likes to impress you, by sending you better and better applicants
- If you can get access to the list of applicants in advance, you can create your own randomization, by randomly permuting the list and then interviewing the applicants.
- Move from (passive) probabilistic analysis to (active) randomized algorithm by putting the randomization under your control!

- Instead of relying on a (perhaps incorrect) assumption that inputs exhibit some distribution, make your own input distribution by, say, permuting the input randomly or taking some other random action
- On the same input, a randomized algorithm has multiple possible executions
- No one input elicits worst-case behavior
- Typically we analyze the average case behavior for the worst possible input

- Suppose we have access to the entire list of candidates in advance
- Randomly permute the candidate list
- Then interview the candidates in this random sequence
- Expected number of hirings/firings is O(log n) *no matter what the original input is*

Probabilistic Analysis versus Randomized Algorithm

- Probabilistic analysis of a deterministic algorithm:
  - assume some probability distribution on the inputs
- Randomized algorithm:
  - use random choices in the algorithm

Generating Random Permutations

How to Randomly Permute an Array

- input: array A[1..n]
- for i := 1 to n do
  - j := value in [i..n] chosen uniformly at random
  - swap A[i] with A[j]

- Show that after i-th iteration of the for loop:

  A[1..i] equals each permutation of i elements from {1,…,n} with probability $(n-i)!/n!$

- **Basis:** After first iteration, A[1] contains each permutation of 1 element from {1,…,n} with probability $(n-1)!/n! = 1/n$

  - true since A[1] is swapped with an element drawn from the entire array uniformly at random

- **Induction:** Assume that after (i–1)-st iteration of the for loop

  A[1..i–1] equals each permutation of i–1 elements from {1,…,n} with probability (n–(i–1))!/n!

- The probability that A[1..i] contains permutation $x_1, x_2, \ldots, x_i$ is the probability that A[1..i–1] contains $x_1, x_2, \ldots, x_{i-1}$ after the (i–1)-st iteration AND that the i-th iteration puts x_i in A[i].

- Let e_1 be the event that A[1..i−1] contains x_1, x_2, …, x_{i-1} after the (i−1)-st iteration.
- Let e_2 be the event that the i-th iteration puts x_i in A[i].
- We need to show that $Pr[e_1 \cap e_2] = (n-i)!/n!$.
- Unfortunately, e_1 and e_2 are not independent: if some element appears in A[1..i −1], then it is not available to appear in A[i].

- Recall: e_1 is event that $A[1..i-1] = x_1,\ldots,x_{i-1}$
- Recall: e_2 is event that $A[i] = x_i$
- $Pr[e_1 \cap e_2] = Pr[e_2|e_1] \cdot Pr[e_1]$
- $Pr[e_2|e_1] = 1/(n-i+1)$ because
  - x_i is available in A[i..n] to be chosen since e_1 already occurred and did *not* include x_i
  - every element in A[i..n] is equally likely to be chosen
- $Pr[e_1] = (n-(i-1))!/n!$ by inductive hypothesis
- So $Pr[e_1 \cap e_2] = [1/(n-i+1)] \cdot [(n-(i-1))!/n!]$

- After the last iteration (the n-th), the inductive hypothesis tells us that

  A[1..n] equals each permutation of n elements from {1,…,n} with probability $(n-n)!/n! = 1/n!$

- Thus the algorithm gives us a uniform random permutation.