Data Structures and Algorithm Analysis (CSC317)

Randomized algorithms

Hiring problem

- We always want the best hire for a job!
- Using employment agency to send one candidate at a time
- Each day, we interview one candidate
- We must decide immediately if to hire candidate and if so, fire previous!

Hiring problem

- Always want the best hire…
- Cost to interview (low)
- Cost to fire/hire… (expensive)

Hire-Assistant(n)

```
1.    best = 0     //least qualified candidate
2.    for i = 1 to n
3.        interview candidate i
4.        if candidate i better than best
5.            best = i
6.            hire candidate i
```

Hiring problem

- Always want the best hire… fire if better candidate comes along…
- Cost to interview (low) c_i
- Cost to fire/hire… (expensive) c_h

$$c_h > c_i$$

n Total number candidates

m Total number hired

$O(?)$

Hiring problem

- Always want the best hire… fire if better candidate comes along…
- Cost to interview (low) c_i
- Cost to fire/hire… (expensive) c_h

$$c_h > c_i$$

n Total number candidates

m Total number hired

Different type of cost – not run time, but cost of hiring

Hire-Assistant(n)

```
1.    best = 0     //least qualified candidate
2.    for i = 1 to n
3.        interview candidate i
4.        if candidate i better than best
5.            best = i
6.            hire candidate i
```

Different type of cost, but flavor of max problems, which candidate is best/winning

Hiring problem

- Always want the best hire… fire if better candidate comes along…
- Cost to interview (low) c_i
- Cost to fire/hire… (expensive) c_h

$$c_h > c_i$$

n Total number candidates

m Total number hired

$$O(c_i n + c_h m)$$

Different type of cost – not run time, but cost of hiring

Hiring problem

- Always want the best hire… fire if better candidate comes along…
- Cost to interview (low) c_i
- Cost to fire/hire… (expensive) c_h

$$c_h > c_i$$

n Total number candidates

m Total number hired

When is this most expensive?

Hiring problem

- Always want the best hire… fire if better candidate comes along…
- Cost to interview (low) c_i
- Cost to fire/hire… (expensive) c_h

n Total number candidates

m Total number hired

When is this most expensive?
When candidates interview in reverse order, worst first…

Hiring problem

1. *best* = 0 //least qualified candidate
2. **for** i = worst_candidate **to** best_candidate…

$$O(c_i n + c_h n)$$

When is this most expensive?
When candidates interview in reverse order, worst first…

Hiring problem

1. *best* = 0 //least qualified candidate
2. **for** i = worst_candidate **to** best_candidate…

$$O(c_i n + c_h n) \overset{c_h > c_i}{=} O(c_h n)$$

When is this most expensive?
When candidates interview in reverse order, worst first…

Hiring problem

- Always want the best hire… fire if better candidate comes along…
- Cost to interview (low) c_i
- Cost to fire/hire… (expensive) c_h

n Total number candidates

m Total number hired

When is this least expensive?

Hiring problem

1. *best* = 0 //least qualified candidate
2. **for** i = best_candidate **to** worst_candidate…

$$O(c_i n + c_h)$$

When is this least expensive?
When candidates interview in order, best first…

Hiring problem

1. *best* = 0 //least qualified candidate
2. **for** i = best_candidate **to** worst_candidate…

$$O(c_i n + c_h)$$

When is this least expensive?
When candidates interview in order, best first…
But we don't know who is good or bad a priori…

Hiring problem

- Cost to interview (low c_i)
- Cost to fire/hire… (expensive c_h)

n Total number candidates

m Total number hired

$$O(c_i n + c_h m)$$

Constant no matter order of candidates

Depends on order of candidates!

Hiring problem and randomized algorithms

$$O(c_i n + c_h m)$$

Depends on order of candidates!

Constant no matter order of candidates

Randomized order can do us well
on average (we will show!)

Hiring problem and randomized algorithms

$$O(c_i n + c_h m)$$

Depends on order of candidates!

- Can't assume in advance that candidates are in random order – there might be bias
- But we can add randomization to our algorithm – by asking the agency to send names in advance, and randomizing

Hiring problem - randomized

- We always want the best hire for a job!
- Employment agency sends list of n candidates in advance
- Each day, we choose randomly a candidate from the list to interview
- We do not rely on the agency to send us randomly, but rather take control in algorithm

Randomized hiring problem(n)

1. randomly permute the list of candidates
2. Hire-Assistant(n)

Including random number generator

What do we mean by randomly permute?

Including random number generator

Random(a,b)

Function that returns an integer between a and b, inclusive, where each integer has equal probability

Most programs: pseudorandom-number generator

Including random number generator

Random(a,b)

Function that returns an integer between a and b, inclusive, where each integer has equal probability

Random(0,1)?

Including random number generator

Random(a,b)

Function that returns an integer between a and b, inclusive, where each integer has equal probability

Random(0,1)?

0 with prob .5
1 with prob .5

Including random number generator

Random(a,b)

Function that returns an integer between a and b,
Inclusive, where each integer has equal probability

Random(3,7)?

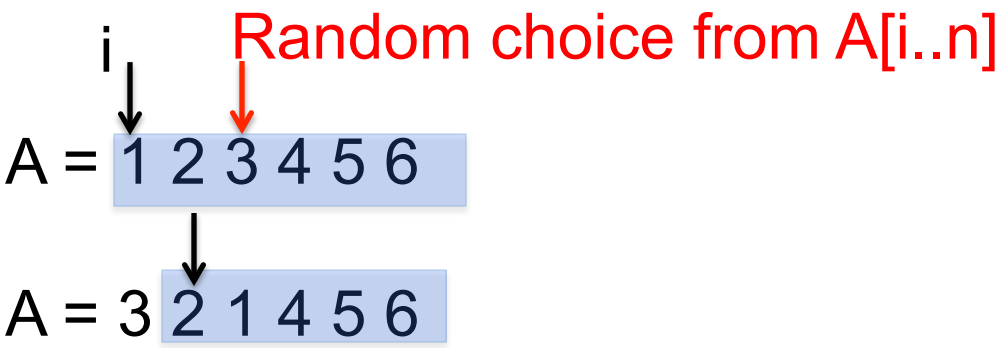
3 with prob 1/5
4 with prob 1/5
5 with prob 1/5
…
Like rolling a (7-3+1) dice

How do we randomly permute?

Random permutation

Randomize-in-place(A,n)

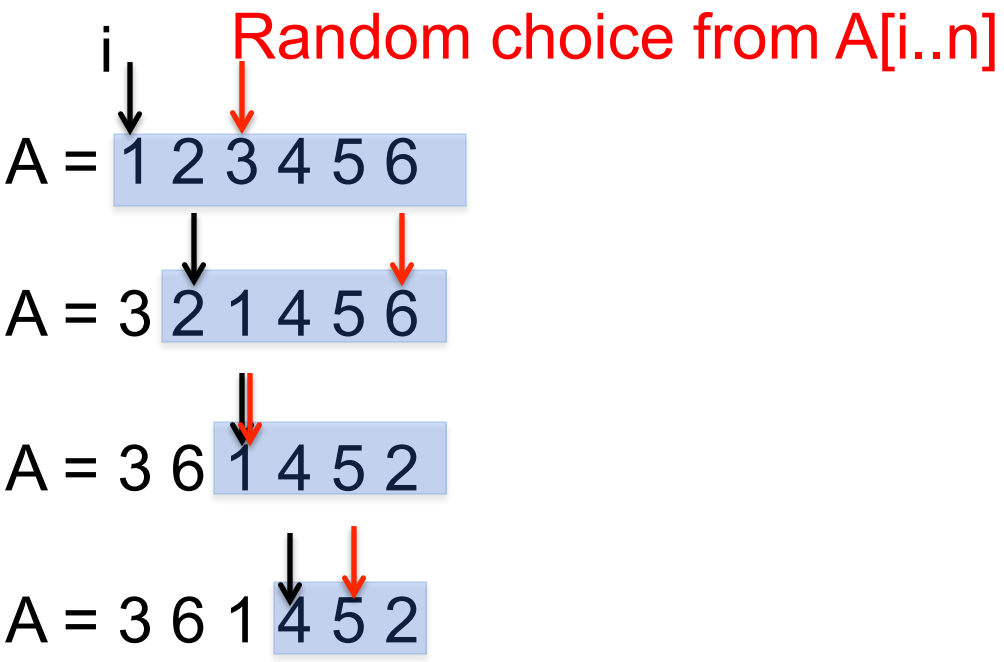
1. For i = 1 to n
2. swap A[i] with A[Random(i,n)]



Random permutation

Randomize-in-place(A,n)

1. For i = 1 to n
2. swap A[i] with A[Random(i,n)]



….