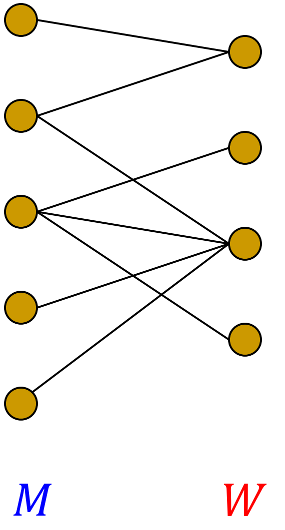
CSC3160: Design and Analysis of Algorithms

Week 10: Stable Matching and Secretary Problem

Instructor: Shengyu Zhang

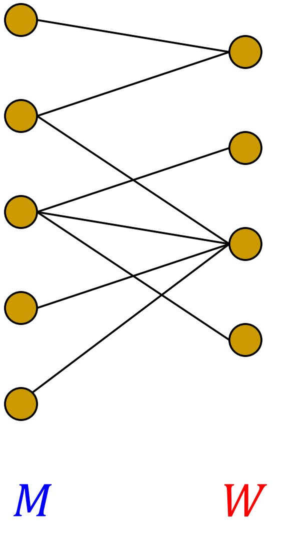
Bipartite graph

- (Undirected) Bipartite graph:
- $G = (V, E)$ for which V can be partitioned into two parts
  - $V = M \cup W$ with $M \cap W = \emptyset$,
- And all edges $e = (m, w)$ have $m \in M$ and $w \in W$.



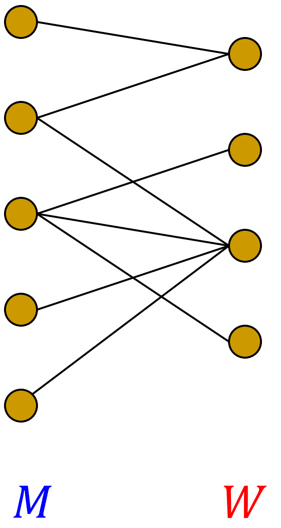
Matching, maximum matching

- Matching: a collection of vertex-disjoint edges
  - a subset $E' \subseteq E$ s.t. no two edges $e, e' \in E'$ are incident.
- $|E'|$: size of matching.
- Maximum matching: a matching with the maximum size.
- This lecture: matching in a bipartite graph



Perfect matching

- There may be some vertices not incident to any edge.
- Perfect matching: a matching with no such isolated vertex.
  - needs at least: $|M| = |W|$
- We'll assume $|M| = |W|$ in the rest of the lecture.



Men's Preference

- Suppose a man sees these women.




- He has a preference among them.
  - What's your preference list?
- Different men may have different lists.

Women's preference

- Women also have their preference lists.




- Assume no tie.
  - The general case can be handled similarly.

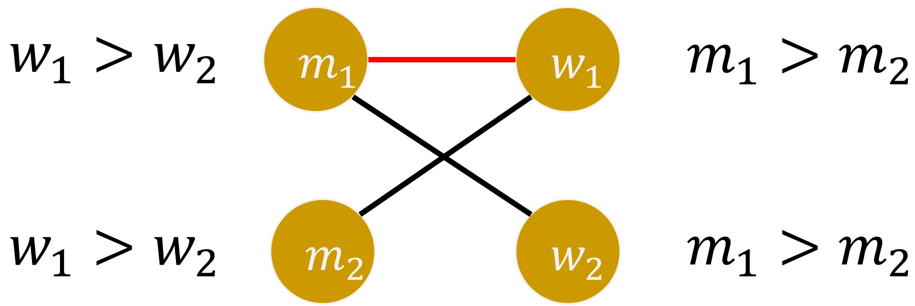
Setting

- n men, n women
- Each man has a preference list of all women
- Each woman has a preference list of all men
- We want to match them.

$w_1 > w_2 > w_3 > w_4$	m_1	w_1	$m_3 > m_1 > m_2 > m_4$
$w_1 > w_2 > w_3 > w_4$	m_2	w_2	$m_3 > m_4 > m_1 > m_2$
$w_2 > w_1 > w_3 > w_4$	m_3	w_3	$m_1 > m_4 > m_2 > m_3$
$w_3 > w_2 > w_4 > w_1$	m_4	w_4	$m_4 > m_1 > m_3 > m_2$

A stability property

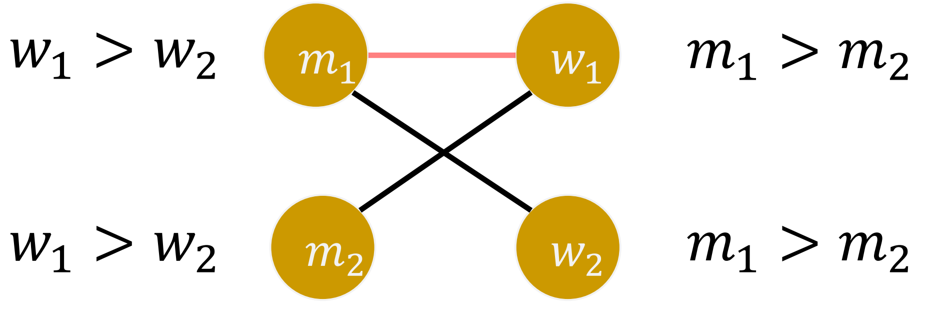
- Suppose there are two couples with these preferences.



- The marriage is unstable, because m_1 and w_1 like each other more than their currently assigned ones!

Stability

- Such a pair is called a blocking pair.



- *Question: Can we have a matching without any blocking pair?*
  - Such a matching is then called a stable matching.

Real applications

- If you think marriage is a bit artificial since there is no centralized arranger, here is a real application.
- Medical students work as interns at hospitals.
  - In the US more than 20,000 medical students and 4,000 hospitals are matched through a clearinghouse, called NRMP (National Resident Matching Program).

Real applications

- Students and hospitals submit preference rankings to the clearinghouse, who uses a specified rule to decide who works where.

- *Question: What is a good way to match students and hospitals?*

More than one question

- *Question: Does a stable matching always exist?*
- *Question: If yes, how to find one?*
- *Question: What mathematical / economic properties it has?*

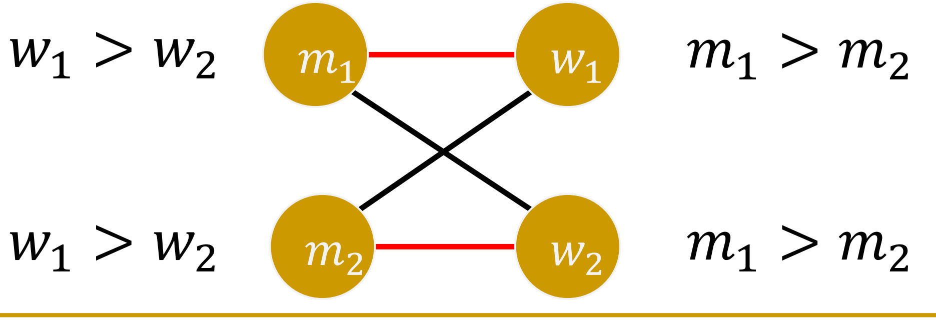
Good news: Stable matchings always exist.

- Theorem (Gale-Shapley) For any given preference lists, there always exists a stable matching.

- They actually gave an algorithm, which bears some resemblance to real marriages.

Consider a simple dynamics

- $\forall$ matching f, $\forall$ blocking pair (m, w),
  - Remove the old pairing $\big(m, f(m)\big)$ and $\big(w, f(w)\big)$
    - $f(m)$: the woman matched to m in f. ($f(w)$: similar.)
  - Match m and w
  - Match $f(m)$ and $f(w)$
- Question: Would repeating this finally lead to a stable matching?



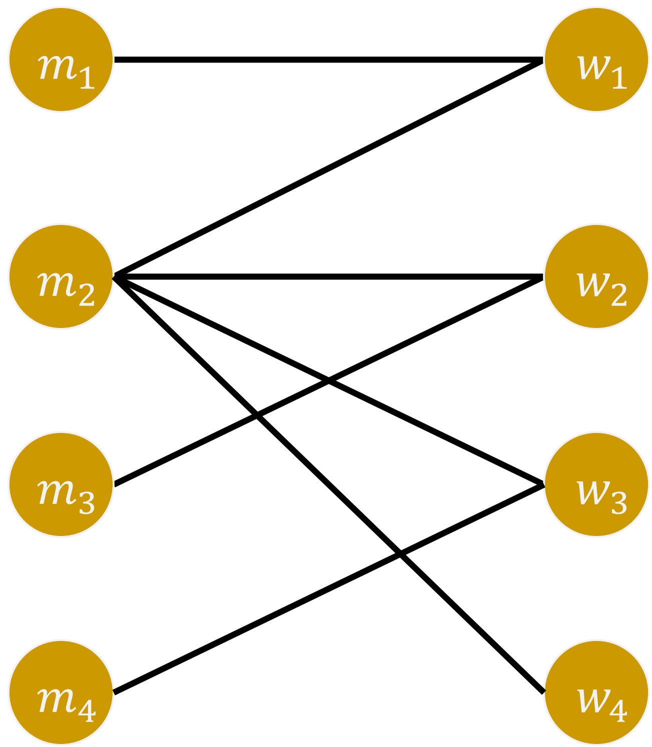
Example

- Can you find an counterexample?

- Next we'll give an algorithm that actually works.
- Let's first run the algorithm on an example.

Algorithm by an example

$w_1 > w_2 > w_3 > w_4$	m_1	w_1	$m_3 > m_1 > m_2 > m_4$
$w_1 > w_2 > w_3 > w_4$	m_2	w_2	$m_3 > m_4 > m_1 > m_2$
$w_2 > w_1 > w_3 > w_4$	m_3	w_3	$m_1 > m_4 > m_2 > m_3$
$w_3 > w_2 > w_4 > w_1$	m_4	w_4	$m_4 > m_1 > m_3 > m_2$



Gale-Shapley (Deferred-Acceptance) Algorithm

- Initially all men and women are free
- **while** there is a man m who is free and hasn't proposed to every woman
  - choose such a man m arbitrarily
  - let w be the highest ranked woman in m's preference list to whom m hasn't proposed yet
  - *// next: m proposes to w*
  - **if** w is free, **then** (m, w) become engaged
  - **else**, suppose w is currently engaged to m'
    - **if** w prefers m' to m, **then** m remains free
    - **if** w prefers m to m', **then** (m, w) becomes engaged and m' becomes free
- Return the set of engaged pairs as a matching

Analysis of the algorithm

- We will show the following:

1. The algorithm always terminates…
2. … in $O(n^2)$ steps, // n men and n women.
3. and generates a stable matching.

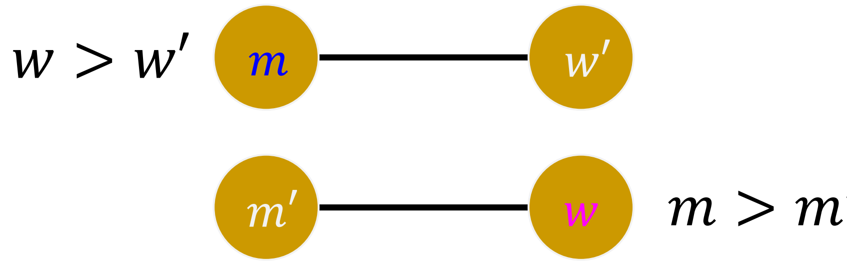
Some observations

- In each iteration, one man m proposes to a new woman w.
- For any man: The women he proposes to get worse and worse
  - according to his preference list
- Because he proposes to a new woman only when the previous one dumps him
  - forcing him to try next (worse!) ones.

Time bound

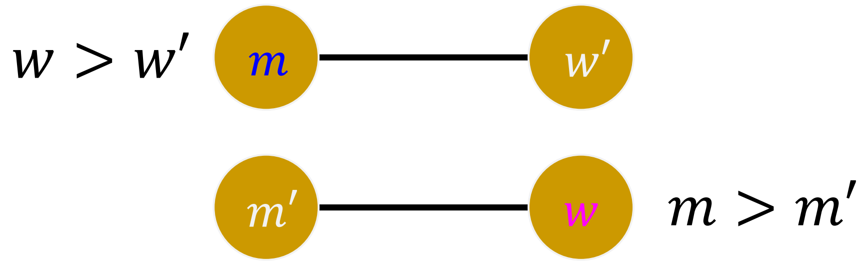
- Each man proposes at most n steps.
  - since his proposed women are worse and worse
- There are n men.
- Therefore: at most n^2 proposals.
- Since each iteration has exactly one proposal, there are at most n^2 iterations.
- Theorem. Gale-Shapley algorithm terminates after at most n^2 iterations.

Correctness



- Suppose the algorithm returns a matching f with a blocking pair (m, w),
  - i.e. m prefers w to w' and w prefers m to m', where w' and m' are their current partner.
- Note: m's last proposal was to w'; see the algorithm.
- m has proposed to w before to w'.
  - Since m proposes from best to worst.
- But at the end of the day, w chose m'
- So m' also proposed to w at some point.

Correctness



- Suppose the algorithm returns a matching f with a blocking pair (m, w),
  - i.e. m prefers w to w' and w prefers m to m', where w' and m' are their current partner.
- So both m and m' proposed to w.
- And w finally married m' instead of m.
- No matter who, m or m', proposed first, w prefers m' to m.
- A contradiction to our assumption.

Some observations

- For any man: His fiancé gets worse and worse (according to his preference list)
  - because he changes fiancé only when the previous one dumps him, forcing him to try next (worse!) ones.
- For any woman: Her fiancé gets better and better (according to her preference list)
  - because she changes fiancé only when a better man proposes to her.

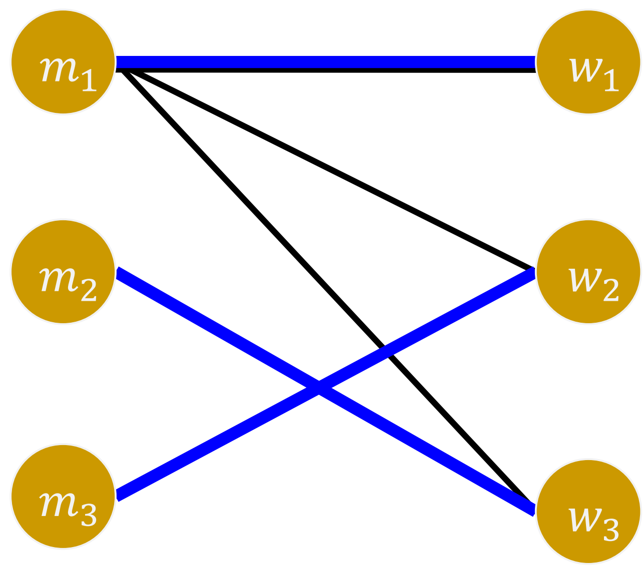
Women propose?

- What if women propose?

$w_1 > w_2 > w_3$

$w_1 > w_2 > w_3$

$w_1 > w_3 > w_2$



$\cancel{m_1} > m_2 > m_3$

$\cancel{m_1} > \cancel{m_3} > m_2$

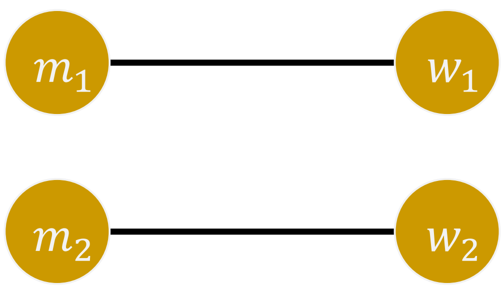
$\cancel{m_1} > \cancel{m_2} > m_3$

Which stable matching is better?

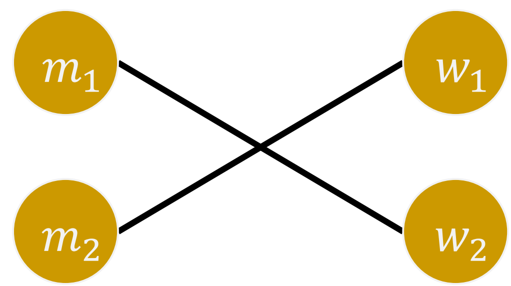
m_1: $w_1 > w_2$ | w_1: $m_2 > m_1$

m_2: $w_2 > w_1$ | w_2: $m_1 > m_2$

GS algorithm: men propose



GS algorithm: women propose



- As a man, which matching you prefer?
  - What if you are m_1? What if you are m_2?
- As a woman, which matching you prefer?
  - What if you are w_1? What if you are w_2?

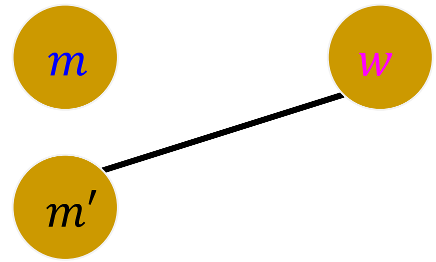
Stable Matching by G-S, men propose

- For any man m, his set of valid partners is
  $vp(m) = \{w\colon f(m) = w \text{ for some stable matching } f\}$
- $best(m)$: the best $w \in vp(m)$.
  - "best": according to m's preference.
- Theorem. Gale-Shapley algorithm matches all men m to $best(m)$.
- Implications:
  - different orders of free men picked do not matter
  - for any men $m_1 \neq m_2$, $best(m_1) \neq best(m_2)$

Proof

- For contradiction, assume that some m^* is matched to worse than $w^* = best(m^*)$.
- Since m^* proposes in the decreasing order, m^* must be rejected by w^* in the course of the GS algorithm.
- Note that $w^* \in vp(m^*)$. So there exists a man rejected by his valid partner.

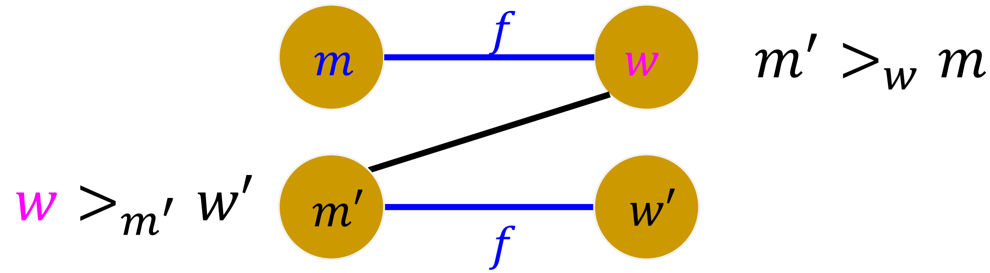
Proof



$m' >_w m$

- Consider the first such moment t that some m is rejected by some $w \in vp(m)$.
- Since m proposes in the decreasing order, $w = best(m)$.
- What triggers the rejection?
  - Either m proposed but was turned down (w prefers her current partner),
  - or w broke her engagement to m in favor of a better proposal.
- In either case, at moment t, w is engaged to a man m' whom she prefers to m, i.e., $m' >_w m$.

Proof



- By def of $best(m)$, ∃ a stable matching f assigning m to w.
- Assume that m' is matched to $w' \neq w$ in f.
- At moment t, m is *first* man rejected by someone in $vp(m)$.
- So no one in $vp(m')$, including w', rejected m' by now.
  - $w' \in vp(m')$ since w' and m' are paired up in the stable matching f.
- If $w <_{m'} w'$, m' should have proposed to w'. But now m' is with w, so m' has been dumped by w'. Impossible.
- Hence $w >_{m'} w'$. Contradiction to fact that f is stable. □

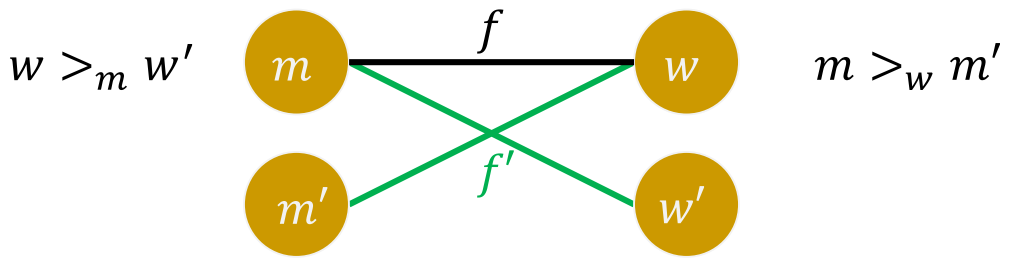
How about women?

- Recall: $best(m)$ is the best woman matched to m in all possible stable matchings.
- GS algorithm matches all men m to $best(m)$.
- $worst(w)$ is the worst man matched to w in all possible stable matchings.

- Theorem. GS algorithm matches all women w to $worst(w)$.

Proof

- By the last theorem, each m is matched to $w = best(m)$ when GS(men propose) gives f.
- We'll show that $m = worst(w)$.
- Suppose there is a stable matching f' in which w is matched to an even worse $m' <_w m$.
- Consider m's partner in f'; call her w'.
- $w >_m w'$, because $w = f(m) = best(m)$.
- Then (m, w) is a blocking pair in f'. Contradiction!



Who should propose?

- Thus if men propose, then
- in each man's eyes:
  - His engaged women get worse and worse.
  - But finally he gets the best possible. (The best that avoids a later divorce.)
- in each woman's eyes:
  - Her engaged men get better and better.
  - But finally she gets the worst possible. (The worst that avoids a later divorce.)

Summary for Stable Matching

- A bipartite matching is stable if no block pair exists.
- Gale-Shapley algorithm finds a stable matching by at most n^2 iterations.
- Whichever side proposes finally get their best possible.

Secretary hiring problem

When to settle down?

- Continuing the discussion about “marriage”, a related problem is:

  When to settle down?

- *Secretary problem*:
  - We want to hire a new office assistant.
  - There are a number of candidates.
  - We can interview one candidate each day, but we have to decide the acceptance/rejection immediately.

One possible strategy

- On each day, if candidate A is better than the current secretary B, then fire B and hire A.
  - Each has a score. Assume no tie.
- Firing and hiring always have overhead.
  - Say: cost c.
- We'd like to pay this but it'll be good if we could have an estimate first.
- *Question: Assuming that the candidates come in a random order, what's the expected total cost?*

Probability…

- Define a random variable X

  $X =$ # of times we hire a new secretary

- Our question is just to compute
$$\mathbf{E}[cX] = c \cdot \mathbf{E}[X].$$
- By definition,
$$\mathbf{E}[X] = \sum_{x=1}^{n} x \cdot \mathbf{Pr}[X = x].$$
- But this seems complicated to compute.

Indicator variables

- Now we see how to compute it easily, by introducing some new random variables.
- Define $X_i = \begin{cases} 1 & \text{if candidate } i \text{ has been hired} \\ 0 & \text{otherwise} \end{cases}$.
- Then $X = \sum_{i=1}^{n} X_i$.
- Recall the linearity of expectation:

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

- We thus have $\mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i]$.

Analysis continued

- What is $\mathbf{E}[X_i]$?
- Recall $X_i = \begin{cases} 1 & \text{if candidate } i \text{ has been hired} \\ 0 & \text{otherwise} \end{cases}$.
- Thus $\mathbf{E}[X_i] = \mathbf{Pr}[X_i = 1] = 1/i$.
  - Candidate i was hired iff she is the best among the first i candidates.
- So $\mathbf{E}[X] = \sum_{i=1}^{n} \mathbf{E}[X_i] = \sum_{i=1}^{n} 1/i \approx \ln(n)$.
- The average cost is $\ln(n) \cdot c$.

Another strategy

- A more natural scenario is that we only hire once.
- And of course, we hope to hire the best one.
- But the candidates on the market also get other offers. So we need to issue offer fast.
- Interview one candidate each day, and decide acceptance/rejection immediately.
- The candidates come in a random order.

Strategy

- Reject the first k candidates no matter how good they are.
  - Because there may be better ones later.
- After this, hire the first one who is better than all the first k candidates.
- If all the rest $n - k$ are worse than the best one among the first k, then hire the last one.

Pseudo-code

- $best\_score = 0$
- **for** $i = 1$ **to** k

    if $score(i) > best\_score$

        $best\_score = score(i)$

  for $i = k + 1$ **to** n

    if $score(i) > best\_score$

        return(i)

  return n

Next

- We want to determine, for each k, the probability that we hire the best one.
- And then maximize this probability over all k.
- Suppose we hire candidate i.
  - $i > k$ in the strategy (since we choose to reject the first k candidates).
- S: event that we hire the best one.
- S_i: event that we hire the best one, which is candidate i.
- $\mathbf{Pr}[S] = \sum_{i=k+1}^{n} \mathbf{Pr}[S_i]$.

- S_i: candidate i is the best among the n candidates, …
  - probability: $1/n$.
- and candidates $k+1, \dots, i-1$ are all worse than the best one among $1, \dots, k$.
  - so that candidates $k+1, \dots, i-1$ are not hired.
  - probability: $k/(i-1)$. (*The best one among the first* $i-1$ *appears in the first* k.)

Putting together

- $\mathbf{Pr}[S_i] = \frac{1}{n} \cdot \frac{k}{i-1} = \frac{k}{n(i-1)}$.
- So $\mathbf{Pr}[S] = \sum_{i=k+1}^{n} \mathbf{Pr}[S_i]$

$$= \sum_{i=k+1}^{n} \frac{k}{n(i-1)}$$

$$= (k/n) \sum_{i=k}^{n-1} (1/i)$$

$$\approx (k/n)(\ln(n) - \ln(k)).$$

- Maximize this over all $k \in \{1, \ldots, n\}$ we get

$$k = n/e \approx 0.368 \cdot n$$

  - take derivative with respect to k, and set it equal to 0.
- And the success probability is $1/e \approx 0.368$.

Summary for the Secretary problem

- In the first strategy (*always hire a better one*) we hire around $\ln(n)$ times (in expectation).

- In the second strategy (*hire only once*) we hire the best with probability ≈ 0.368.
  - Reject the first $k = 0.368 \cdot n$ candidates
  - And in the rest hire the first one who beats all the first k ones.