Greedy Algorithms

CSE 101: Design and Analysis of Algorithms

Lecture 10

CSE 101: Design and analysis of algorithms

- Greedy algorithms
  - Reading: Kleinberg and Tardos, sections 4.1, 4.2, and 4.3
- Homework 4 is due today, 11:59 PM
- Homework 5 will be assigned today
  - Due Nov 6, 11:59 PM

Techniques to prove optimality

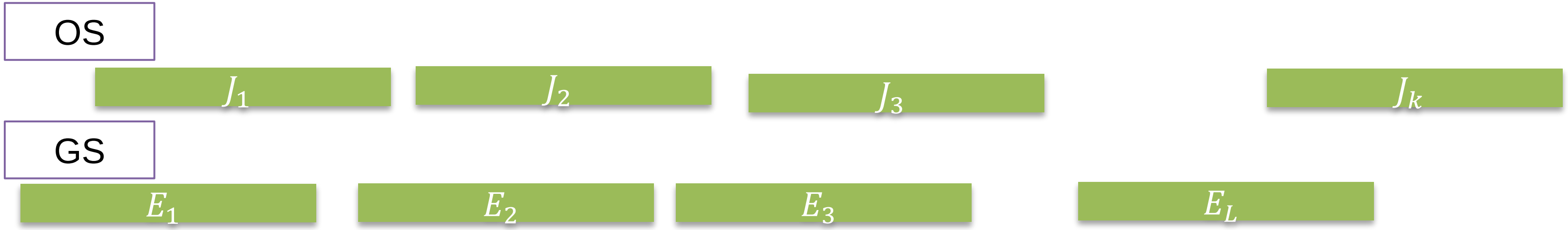
- We will look at a number of general methods to prove optimality
  - Greedy modify the solution (also referred to as greedy exchange): most general
  - Greedy stays ahead: more intuitive
  - Greedy achieves the bound: also comes up in approximation, linear programming, network flow
- Which one to use is up to you, but modify the solution applies almost universally. Others can be easier, but only work in special cases.

Problem specification

- Design an algorithm that uses the greedy choice of picking the next available event with the earliest end time
  - Instance: n events $E_1, \ldots, E_n$ each with a start time s and end time f_i $E_i = (s_i, f_i)$
  - Solution format: list of events
  - Constraints: events cannot overlap
  - Objective: maximize the number of events

Greedy stays ahead

- Instead of just first greedy choice, compare all of the greedy algorithm's solution to all of the other algorithm's solution

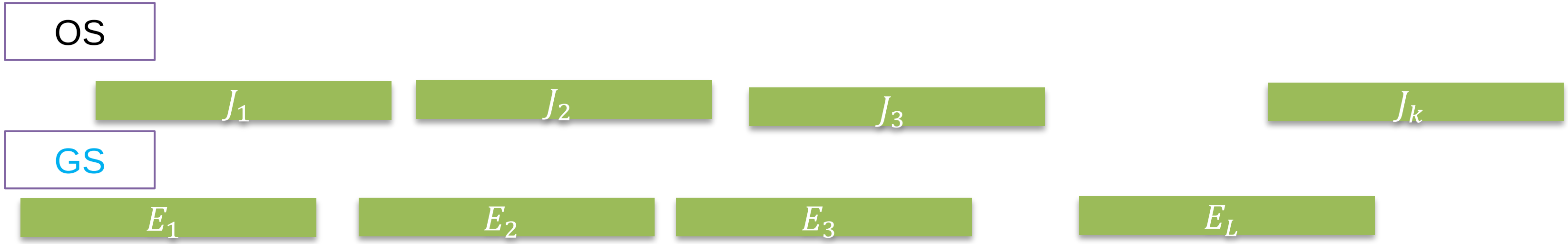


What to show: $L \geq k$, but indirectly by comparing some progress measure of GS to OS

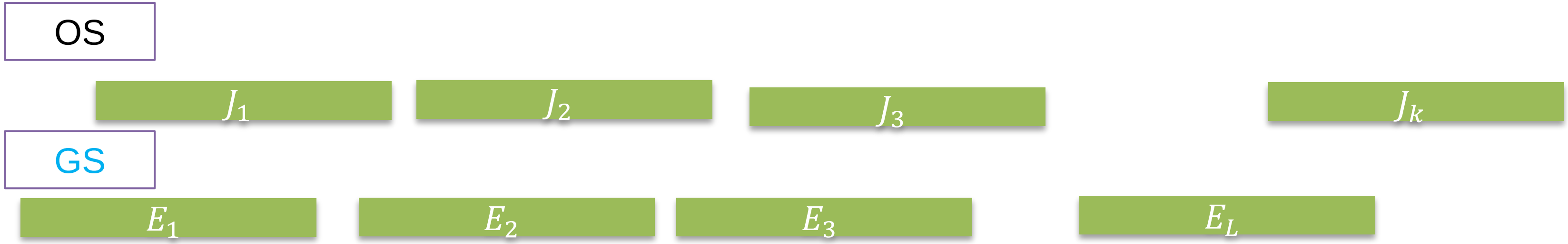
In what way is E1 better than J1, E2 better than J2, etc.?

Greedy stays ahead

- Suppose input is a set of n events $\{A_1, \ldots, A_n\}$ for some $n \geq 1$
- Given: an arbitrary solution OS = $\{J_1, \ldots, J_k\}$ ordered by finish time and greedy solution GS = $\{E_1, \ldots, E_L\}$ ordered by finish time
- Claim: Finish(E_i) ≤ Finish(J_i) for all $i \geq 1$



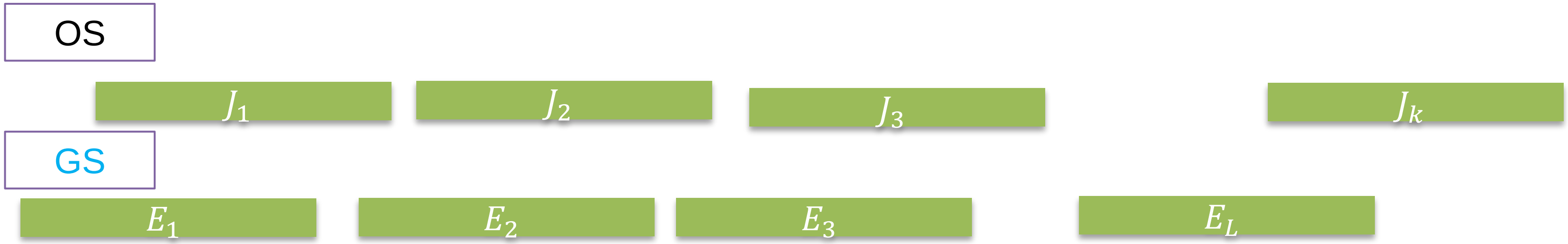
Greedy stays ahead



Claim: Finish (E_i) ≤ Finish(J_i) for all $i \geq 1$

- Base case: Finish(E_1) ≤ Finish(J_1) because of greedy choice
- Inductive hypothesis: Suppose that for some $i \geq 1$, Finish(E_i) ≤ Finish(J_i)
- What to show: Finish(E_{i+1}) ≤ Finish(J_{i+1})

Greedy stays ahead



Claim: Finish (E_i) ≤ Finish(J_i) for all $i \geq 1$

- What to show: Finish(E_{i+1}) ≤ Finish(J_{i+1})
- Finish(J_i) ≤ Start(J_{i+1})
- Finish(E_i) ≤ Finish(J_i), inductive hypothesis
- So, Finish(E_i) ≤ Finish(J_i) ≤ Start(J_{i+1})
- E_{i+1} is the first to finish after Finish(E_i), definition of greedy
  - J_{i+1} is in the set of available events, so Finish(E_{i+1}) ≤ Finish(J_{i+1})

Greedy stays ahead

- Suppose by contradiction that OS has more events than GS
  - |OS| = k, |GS| = L
- In other words, L < k
- E_L is the final greedy choice, so there are no other events that end after E_L that do not conflict with E_L
- By inductive argument: Finish(E_L) ≤ Finish(J_L)
- Finish (J_L) ≤ Start(J_{L+1})
- Then greedy would not end with E_L because J_{L+1} is still available. Contradiction

Greedy stays ahead template

- Define progress measure
- Order the decisions in OS to line up with GS
- Prove by induction that the progress after the *i*-th decision in GS is at least as big as that in OS
- Assume that OS is strictly better than GS
- Use progress argument to arrive at contradiction

Greedy achieves the bound

- This is a proof technique that does not work in all cases
- The way it works is to argue that when the greedy solution reaches its peak cost, it reveals a **bound**
- Then, show this bound is also a lower bound on the cost of any other solution
- So we are showing: Cost(GS) $\leq$ Bound $\leq$ Cost (OS)
- Allows the two inequalities to be separated

Event scheduling with multiple rooms

- Suppose you have a conference to plan with n events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?
- Greedy choice:
  - Number the rooms from 1 to n
  - Sort the events by earliest start time
  - Put the first event in room 1
  - For events 2, …, n, put each event in the smallest numbered room that is available

Event scheduling with multiple rooms

- Suppose you have a conference to plan with n events and you have an unlimited supply of rooms. How can you assign events to rooms in such a way as to minimize the number of rooms?
- Instance: Start and end times of n events
- Solution Format: an assignment of each event to a room
- Constraints: No two events that overlap are assigned to the same room
- Objective: minimize the number of rooms used

Implementation

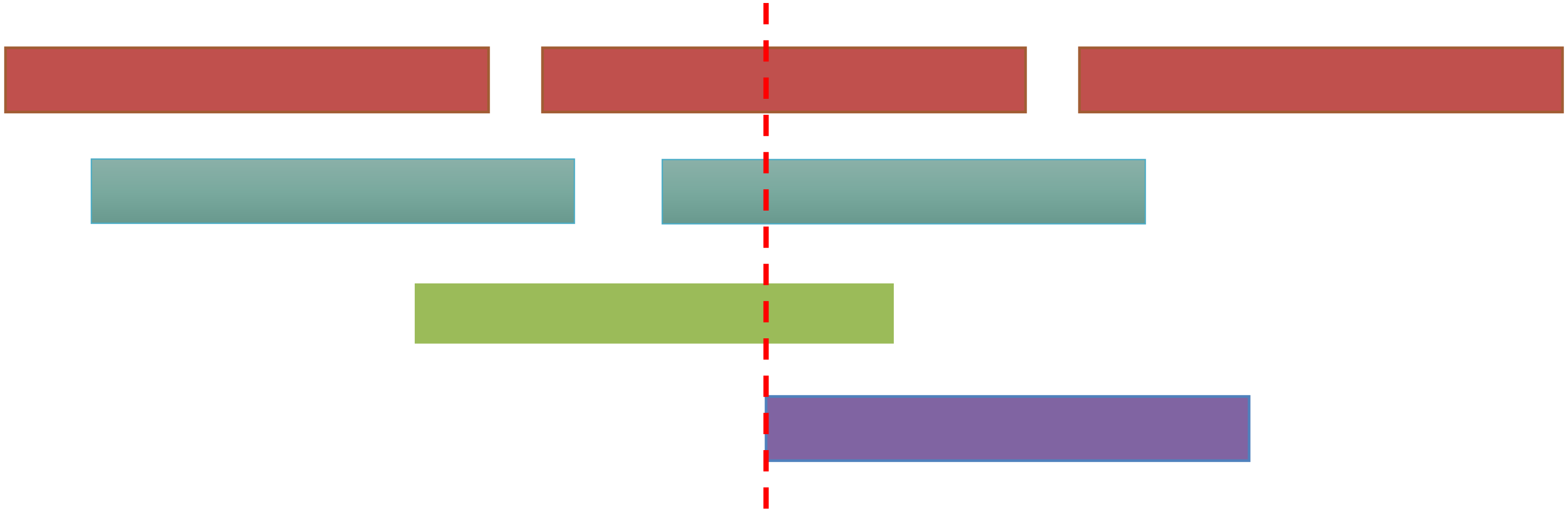
- Sort both start and finish times of events
- Keep priority queue of available rooms ordered by room number
- Go through sorted lists of times
- When an event starts, assign it to the smallest room in the priority queue, and delete that room from the priority queue
- When an event finishes, insert the room it is scheduled in back into the priority queue

- N inserts, N deletes, O(N log N) time to sort: O(N log N) time total

When does GS reach peak cost?

Room

- 1
- 2
- 3
- 4



Why did we have to use four rooms? What happened at the time we reached "peak cost"?

Defining the lower bound

- Let t be a certain time during the conference
- Let $B(t)$ be the set of all events E such that $t \in E$
- Let R be the number of rooms you need for a valid schedule
- Then, $R \geq |B(t)|$ for all t
- Proof
  - For any time t, let $E_{i_1}, \dots, E_{i_{|B(t)|}}$ be all the events in $B(t)$
  - Then, since they all are happening at time t, they all have to be in different rooms, so $R \geq |B(t)|$
- Let $L = \max_t |B(t)|$. Then, L is the lower bound on the number of rooms needed.

Greedy achieves the bound

- Let k be the number of rooms picked by the greedy algorithm. Then, at some point t, $|B(t)| \geq k$ (i.e., there are at least k events happening at time t).
- Proof
  - Let t be the starting time of the first event to be scheduled in room k
  - Then, by the greedy choice, room k was the least number room available at that time
  - This means at time t there was an event happening in room 1, room 2, …, room k-1. And, an additional event happening in room k
  - Therefore, $|B(t)| \geq k$ at some point t

Conclusion: greedy is optimal

- The greedy algorithm uses the minimum number of rooms
  - Let GS be the greedy solution, k = Cost(GS) the number of rooms used in the greedy solution
  - Let k be the number of rooms the greedy algorithm uses and let R be any valid schedule of rooms. There exists a t such that at all time, k events are happening simultaneously. So R uses at least k rooms. So, R uses at least as many rooms as the greedy solution. Therefore, the greedy solution is optimal.

Conclusion: greedy is optimal

- Let GS be the greedy solution, k = Cost(GS) the number of rooms used in the greedy solution
- Let OS be any other schedule, R= Cost(OS) the number of rooms used in OS
- By the bounding lemma, $\mathrm{R} \geq L = \max_t |B(t)|$
- By the achieves the bound lemma, k = |B(t)| ≤ L for some t
- Putting the two together, Cost(GS) =k ≤ R= Cost(OS)
- Thus, the greedy solution is optimal

Greedy achieves the bound

- This is a proof technique that does not work in all cases
- The way it works is to argue that when the greedy solution reaches its peak cost, it reveals a **bound**
- Then, show this bound is also a lower bound on the cost of any other solution
- So we are showing: Cost(GS) ≤ Bound ≤ Cost (OS)
- Allows the two inequalities to be separated

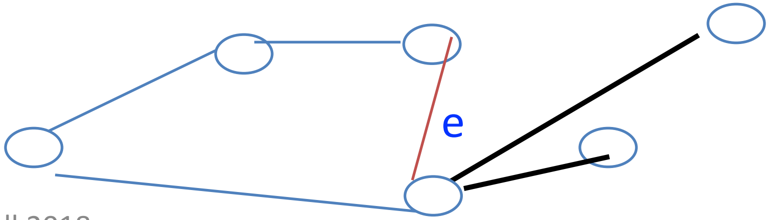
KRUSKAL'S ALGORITHM, PROOF OF CORRECTNESS

General greedy modify the solution template

- Lemma: Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then, there is a solution OS' that does pick g and OS' is at least as good as OS.
  1. State what we know: Definition of g. OS meets constraints.
  2. Define OS' from OS, g   This requires creativity
  3. Prove that OS' meets constraints (use 1, 2)
  4. Compare value/cost of OS' to OS (use 2, sometimes 1)

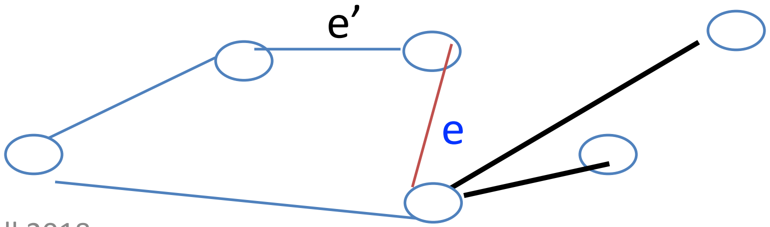
Correctness proof, greedy modify the solution

- The first greedy choice is the smallest weight edge. Let e be the smallest weight edge and let OT be any spanning tree that does not contain e.



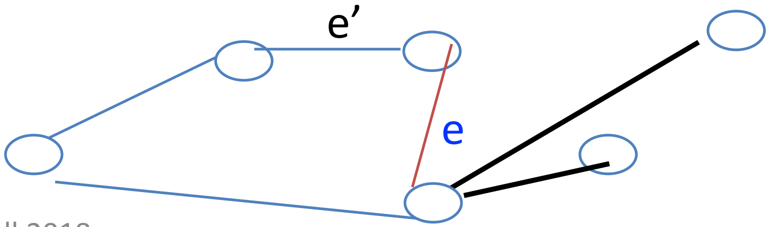
Correctness proof, greedy modify the solution

- The first greedy choice is the smallest weight edge. Let e be the smallest weight edge and let OT be any spanning tree that does not contain e.
- Construct OT' by adding e to OT then removing any other edge e' in the cycle that was created



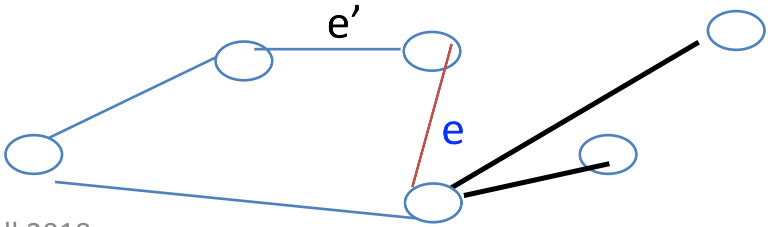
Correctness proof, greedy modify the solution

- The first greedy choice is the smallest weight edge. Let e be the smallest weight edge and let OT be any spanning tree that does not contain e.
- Construct OT' by adding e to OT then removing any other edge e' in the cycle that was created
- We must show that
  1. OT' is a spanning tree
  2. $w(OT') \leq w(OT)$



Correctness proof, greedy modify the solution

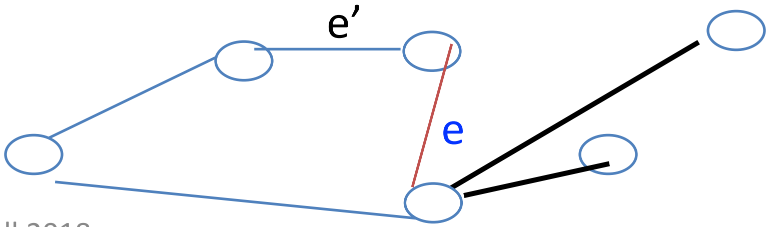
- We must show that
  1. OT' is a spanning tree
  2. $w(OT') \leq w(OT)$



Correctness proof, greedy modify the solution

- We must show that
  1. OT' is a spanning tree
  2. $w(OT') \leq w(OT)$

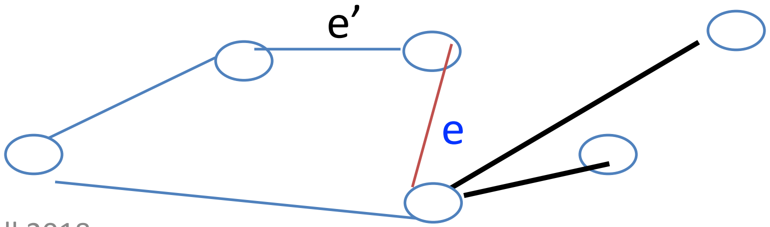
1. We have taken out an edge and added in an edge. We have a graph on n vertices with n-1 edges that does not contain a cycle. Therefore, the graph is a spanning tree.



Correctness proof, greedy modify the solution

- We must show that
  1. OT' is a spanning tree
  2. $w(OT') \leq w(OT)$

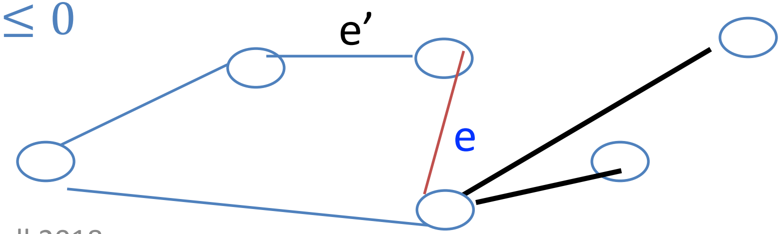
1. We have taken out an edge and added in an edge. We have a graph on n vertices with n-1 edges that does not contain a cycle. Therefore, the graph is a spanning tree.
2. $w(OT') = w(OT) + w(e) - w(e') \leq w(OT)$



Correctness proof, greedy modify the solution

- We must show that
  1. OT' is a spanning tree
  2. $w(OT') \leq w(OT)$

1. We have taken out an edge and added in an edge. We have a graph on n vertices with n-1 edges that does not contain a cycle. Therefore, the graph is a spanning tree.
2. $w(OT') = w(OT) + \underbrace{w(e) - w(e')}_{\leq 0} \leq w(OT)$



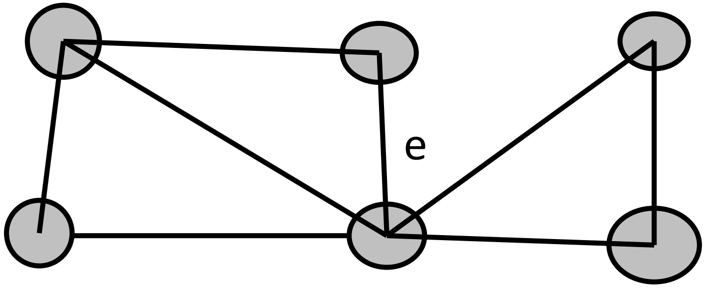
General greedy modify the solution template, induction

- Lemma: Let g be the first greedy decision. Let OS be any legal solution that does not pick g. Then, there is a solution OS’ that does pick g and OS’ is at least as good as OS.
- Prove by strong induction on instance size that *GS* is optimal
- Induction step
  1. Let g be first greedy decision. Let I’ be “rest of problem given g’’
  2. GS = g + GS(I’)
  3. OS is any legal solution
  4. OS’ is defined from OS by the modify the solution argument (if OS does not include g)
  5. OS’ = g + some solution on I’
  6. Induction: GS(I’) at least as good as some solution on I’
  7. GS is at least as good as OS’, which is at least as good as OS

Induction step

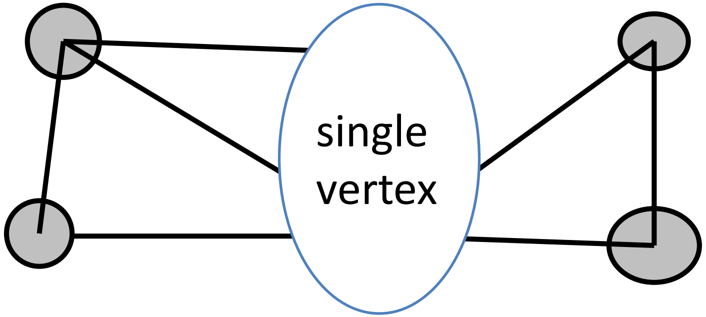
- If G has at most two vertices, any solution is optimal
- Assume Kruskal’s algorithm is optimal for any graph with n-1 vertices
- Let e be the smallest weight edge
- G’: Contract the edge e in G, treating its two vertices as one vertex

Contraction



Contraction

- Contracted graph is not necessarily simple



Induction on number of vertices

- Base case: Kruskal's algorithm generates a minimum spanning tree on any graph with at most two vertices
- Inductive Hypothesis: Suppose Kruskal's algorithm generates a minimum spanning tree for every graph of k vertices for some $k \geq 2$

Induction on number of vertices

- Inductive step: Let G be an arbitrary graph with k+1 vertices. Let OT be any spanning tree of G.
- Then, by the greedy modify the solution lemma, there exists a spanning tree OT' that uses the lightest edge e and $w(OT') \leq w(OT)$
- Contract the two endpoints of e into one vertex and call the resulting graph G'
- Then, since G' has k vertices, kruskal(G') will generate a minimum spanning tree of G

$$w(OT) \geq w(OT') = w(e) + w\big(S(G')\big) \geq w(e) + w(\text{kruskal}(G')) = w(\text{kruskal}(G))$$

Greedy modify the solution lemma

By definition of OT'

By inductive hypothesis

By definition of Kruskal's algorithm

Next lecture

- Greedy algorithms
  - Reading: Kleinberg and Tardos, sections 4.1, 4.2, and 4.3
- Divide and conquer algorithms
  - Reading: Sections 2.1 and 2.2