Lecture 21

CSE 331

Oct 18, 2019

HW 5 grading delayed

Postponed at least till next Friday

Mid-term grading will take precedence

Mini Project updates

To be released over the weekend

Video submission

Problem 2 of coding project

(Hopefully) Problem 3 of coding project

Story behind HWs

note

stop following 35 views

ACM ICPC contest in UB on Oct 26th!

Sorry, this a bit late but wanted to let y'all know that the we will have a WNY preliminary contest for ACM ICPC at UB on Oct 26:

https://cse.buffalo.edu/icpc/

We can have up to 3 teams from UB to compete and we have 2-5 extra spaces in case you are interested in participating in one of the UB teams.

If you are interested in participating, **please email me by 5pm tomorrow (Friday)**.

If you do not to participate but still want to get a sense for the competition, **please volunteer to help out on the 26th** (we need about 5 volunteers). Please email me if you are interested.

logistics

edit · good note | 0

Updated 5 hours ago by Atri Rudra

ACM ICPC contest!

Western NY preliminary contest for ACM ICPC

October 26, 2019

note

stop following 35 views

ACM ICPC contest in UB on Oct 26th!

Sorry, this a bit late but wanted to let y'all know that the we will have a WNY preliminary contest for ACM ICPC at UB on Oct 26:

https://cse.buffalo.edu/icpc/

We can have up to 3 teams from UB to compete and we have 2-5 extra spaces in case you are interested in participating in one of the UB teams.

If you are interested in participating, **please email me by 5pm tomorrow (Friday)**.

If you do not to participate but still want to get a sense for the competition, **please volunteer to help out on the 26th** (we need about 5 volunteers). Please email me if you are interested.

logistics

edit · good note | 0

Updated 5 hours ago by Atri Rudra

Minimum Spanning Tree Problem

Input: Undirected, connected $G = (V,E)$, edge costs c_e

Output: Subset $E' \subseteq E$), s.t. $T = (V,E')$ is connected
$C(T)$ is minimized

If all $c_e > 0$, then T is indeed a tree

Kruskal's Algorithm

Joseph B. Kruskal

Input: $G=(V,E)$, $c_e > 0$ for every e in E

$T = \emptyset$

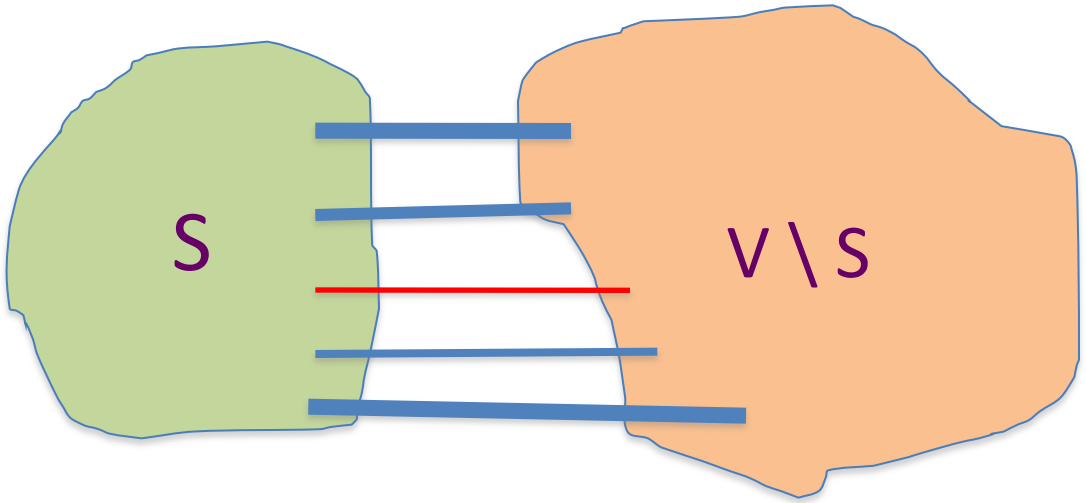
Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

Cut Property Lemma for MSTs

Condition: S and V\S are non-empty



Cheapest crossing edge is in

Did an incorrect "proof" last Friday

Assumption: All edge costs are distinct

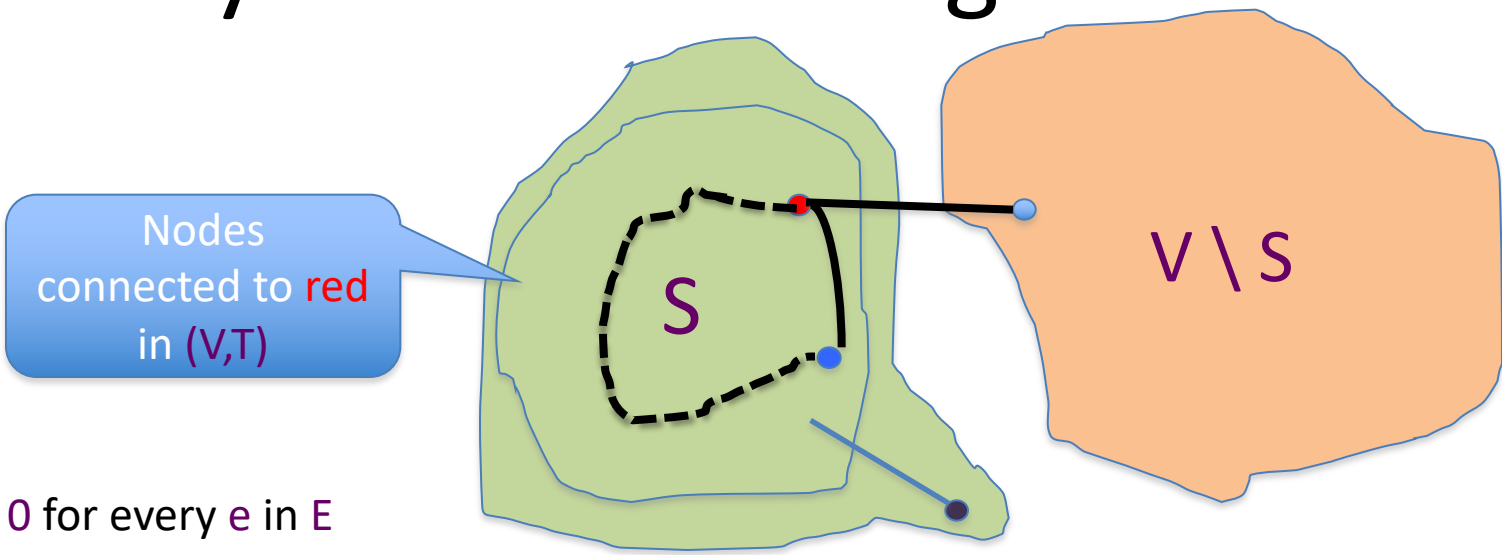
Today's agenda

Prove Cut Property Lemma

Optimality of Kruskal's algorithm

Remove distinct edge weights assumption

Optimality of Kruskal's Algorithm



Input: G=(V,E), $c_e > 0$ for every e in E

T = Ø

Sort edges in increasing order of their cost

Consider edges in sorted order

If an edge can be added to T without adding a cycle then add it to T

S is non-empty

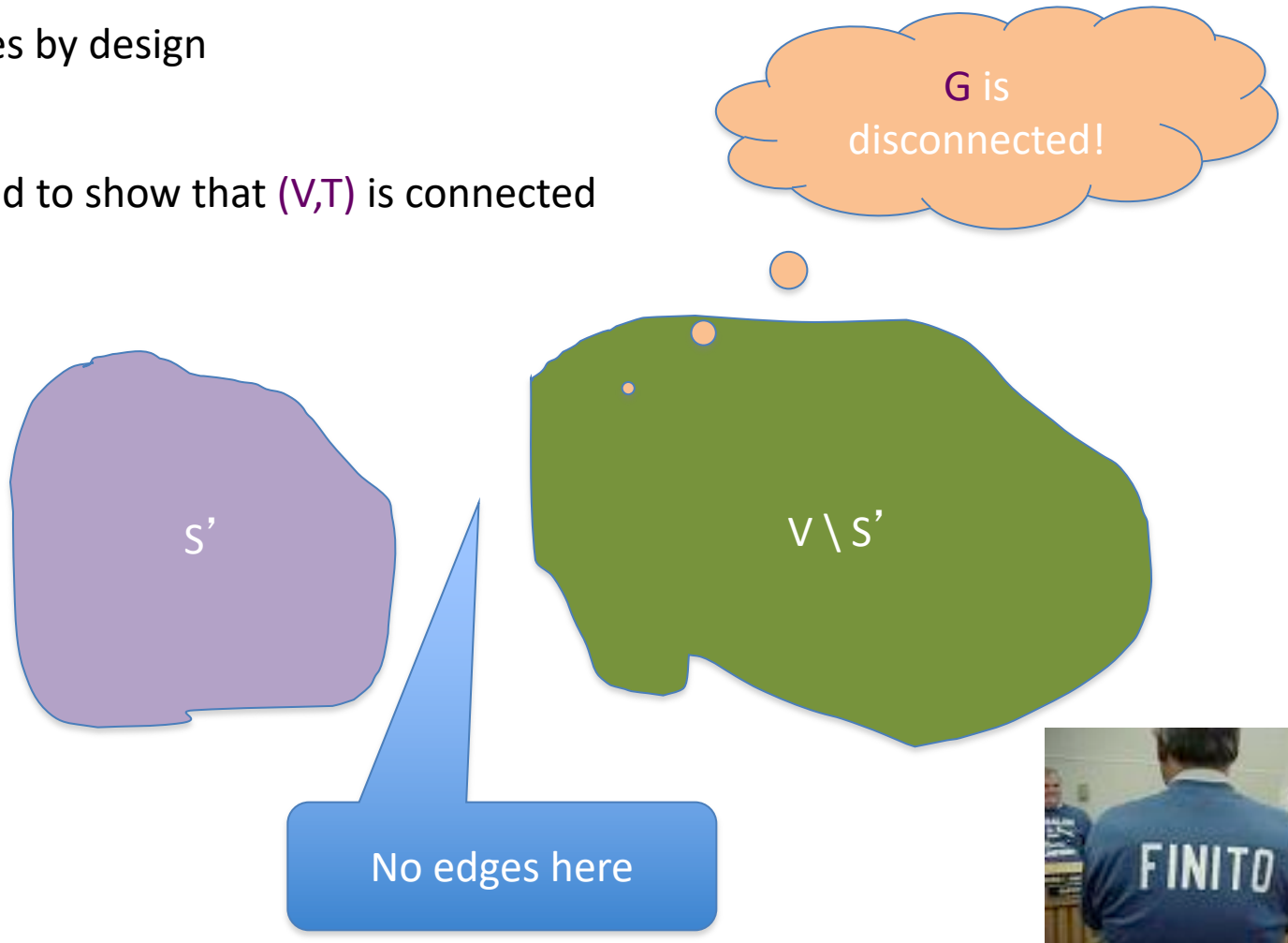
V\S is non-empty

First crossing edge considered

Is (V,T) a spanning tree?

No cycles by design

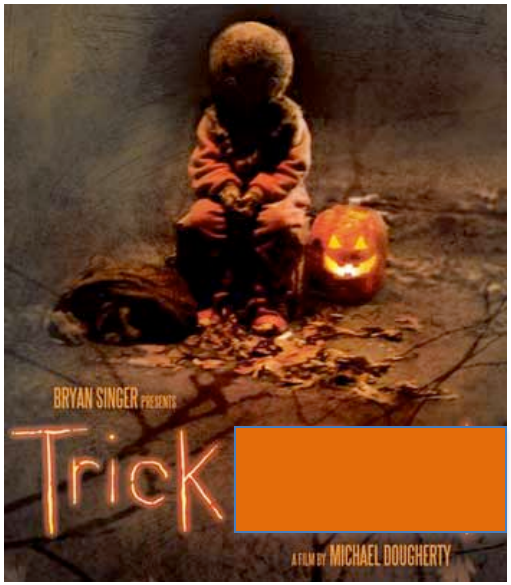
Just need to show that (V,T) is connected



Removing distinct cost assumption

Change all edge weights by very small amounts

Make sure that all edge weights are distinct



MST for "perturbed" weights is the same as for original

Changes have to be small enough so that this holds

EXERCISE: Figure out how to change costs

Running time for Prim's algorithm

Similar to Dijkstra's algorithm

O(m log n)

Input: $G=(V,E)$, $c_e > 0$ for every e in E

$S = \{s\}$, $T = \emptyset$

While S is not the same as V

Among edges $e= (u,w)$ with u in S and w not in S, pick one with minimum cost

Add w to S, e to T

Running time for Kruskal's Algorithm

Can be implemented in O(m log n) time (Union-find DS)

Joseph B. Kruskal

Input: G=(V,E), $c_e > 0$ for every e in E

$O(m^2)$ time overall

T = Ø

Sort edges in increasing order of their cost

Consider edges in sorted order

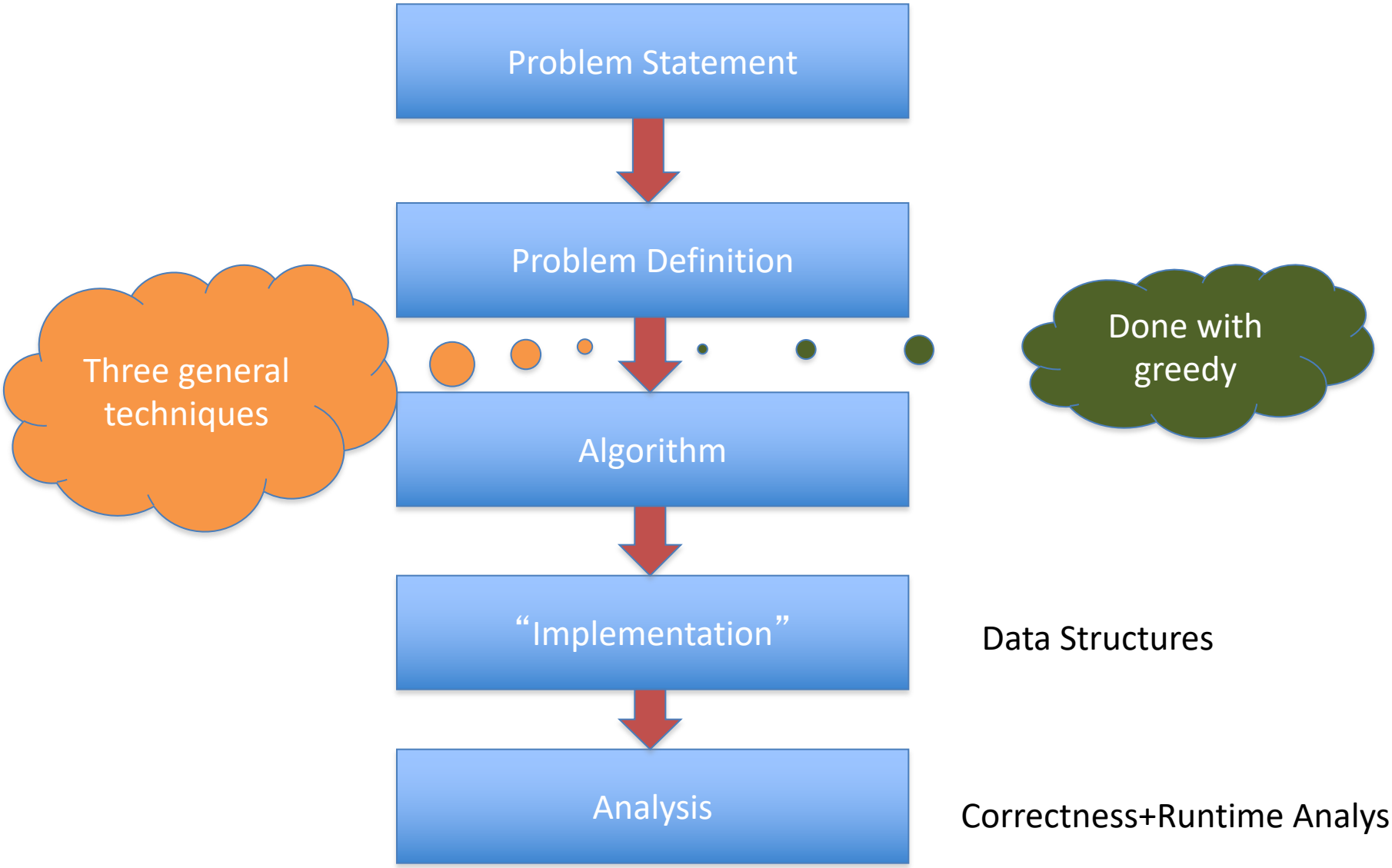
If an edge can be added to T without adding a cycle then add it to T

Can be verified in O(m+n) time

Reading Assignment

Sec 4.5, 4.6 of [KT]

High Level view of the course



Trivia

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

“Patch up” the solutions to the sub-problems for the final solution

Sorting

Given n numbers order them from smallest to largest

Works for any set of elements on which there is a total order

Insertion Sort

Input: $a_1, a_2, \ldots, a_n$

Output: $b_1, b_2, \ldots, b_n$

$O(n^2)$ overall

Make sure that all the processed numbers are sorted

$b_1 = a_1$

for $i = 2 \ldots n$

Find $1 \le j \le i$ s.t. a_i lies between b_{j-1} and b_j

Move b_j to b_{i-1} one cell "down"

$b_j = a_i$

O(log n)

O(n)

a	b
4	4
3	3
2	2
1	1

Other $O(n^2)$ sorting algorithms

Selection Sort: In every round pick the min among remaining numbers

Bubble sort: The smallest number “bubbles” up

Divide and Conquer

Divide up the problem into at least two sub-problems

Recursively solve the sub-problems

“Patch up” the solutions to the sub-problems for the final solution

Mergesort Algorithm

Divide up the numbers in the middle

Sort each half recursively

Merge the two sorted halves into one sorted output

Unless n=2

How fast can sorted arrays be merged?

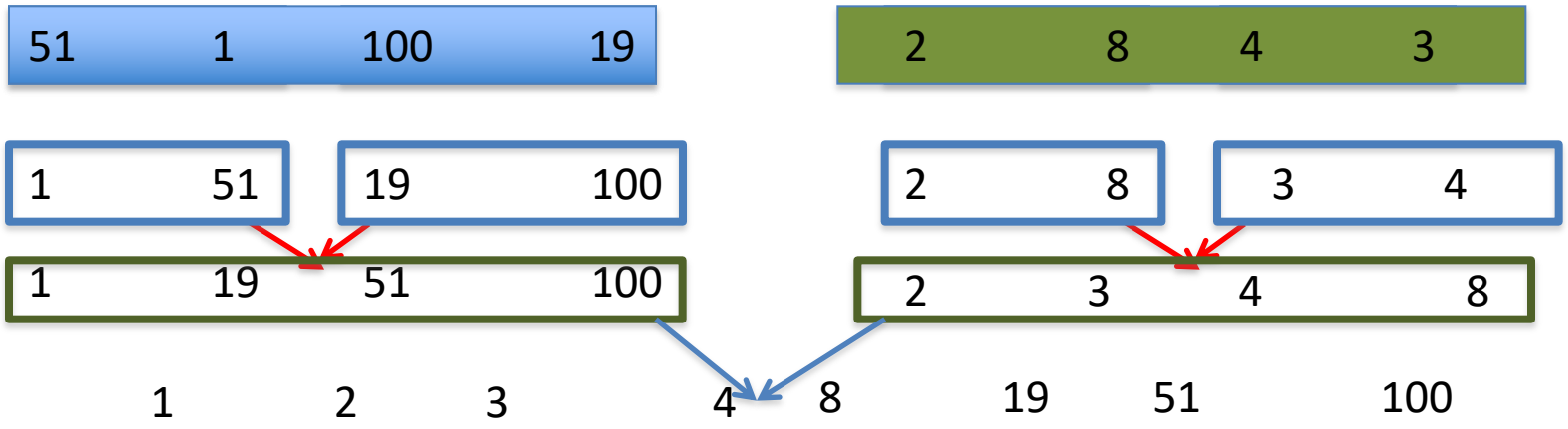
Mergesort algorithm

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order

```
MergeSort( a, n )

    If n = 1 return the order a_1
    If n = 2 return the order min(a_1,a_2); max(a_1,a_2)

    a_L = a_1,..., a_n/2

    a_R = a_n/2+1,..., a_n

    return MERGE ( MergeSort(a_L, n/2), MergeSort(a_R, n/2) )
```

An example run



MergeSort(a, n)

If n = 1 **return** the order a_1

If n = 2 **return** the order min(a_1,a_2); max(a_1,a_2)

$a_L = a_1,\ldots, a_{n/2}$

$a_R = a_{n/2+1},\ldots, a_n$

return MERGE (MergeSort(a_L, n/2), MergeSort(a_R, n/2))

Correctness

Input: $a_1, a_2, \ldots, a_n$

Output: Numbers in sorted order

```
MergeSort( a, n )

  If n = 1 return the order a_1
  If n = 2 return the order min(a_1,a_2); max(a_1,a_2)

  a_L = a_1,..., a_n/2

  a_R = a_n/2+1,..., a_n

  return MERGE ( MergeSort(a_L, n/2) , MergeSort(a_R, n/2) )
```

By induction on n

Inductive step follows from correctness of MERGE