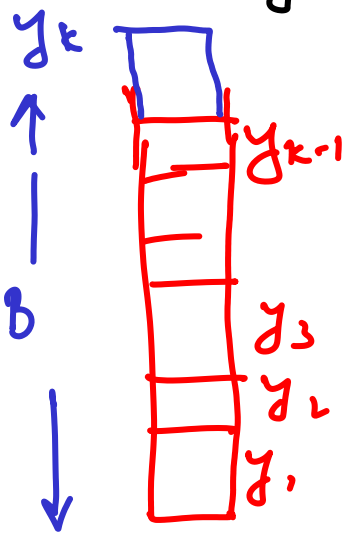
COL 702 Lecture 14 Sept 14

Greedy may not work but may still be effective

Sorting the items in decreasing order of the profit/weight ratios

$y_1, y_2, \ldots y_{k-1} \mid y_k\ y_{k+1} \ldots$



$\max\left[\{y_1, y_2 \ldots y_{k-1}\},\ y_k\right]$

Best possible fractional knapsack soln is $\leq \underbrace{[w(y_1) + w(y_2) + \ldots w(y_{k-1})]}_{A} + \underbrace{[w(y_k)]}_{B}$

OPT of original problem $\leq$ OPT' (fractional knapsack)

$\leq A + B$

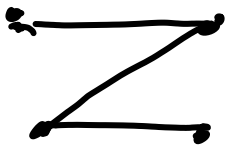
$\Rightarrow$ OPT $\leq 2 \max\{A, B\}$

$\Rightarrow \frac{OPT}{2} \leq \max\{A, B\}$

Guarantee : $\frac{\text{Our soln}}{\text{Best soln}}$

H.W. : Without the term B, construct a counterexample so that only the term A may not be even 10% of OPT

Matching Problem



Greedy can give 50% guarantees

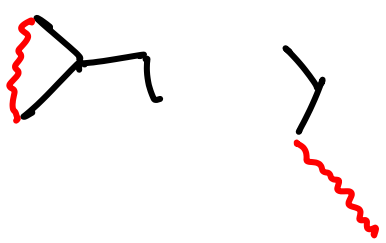
Approximation Algorithm is an important field

Implementation of Kruskal's algorithm

How do you test for cycles?

Observation: If an edge goes across two trees then we can add it else it creates a cycle

Forest



Find If the endpoints (u,v) belong to different trees then we must join T_1, T_2 Union

Find: Given a vertex which tree contains this vertex?

Find $(x) \stackrel{?}{=}$ Find (y)

If Find $(x) \neq$ Find (y) then

Union (Find (x), Find (y))

Union Find data structure

Given subsets $S_1, S_2 \ldots S_k$

we want to maintain a data structure that supports the following operations

Find (x) : returns a set S_i s.t $x \in S_i$

Union (S_1', S_2', S_3') : returns $S_1' \leftarrow S_2' \cup S_3'$

(and implicitly destroys S_2, S_3

We want design a data structure that is efficient for a sequence of union and Find operations

For Kruskals algorithm $2m$ Finds
$n-1$ unions

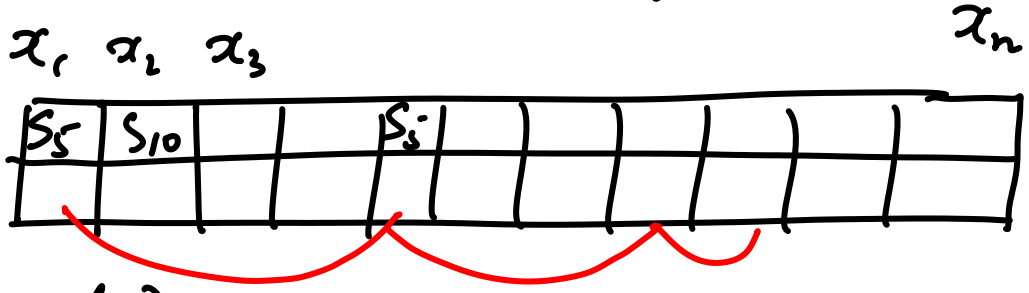
We will focus on disjoint Union Find

Possibilities

1. Each set is a linked list of number elements
   - Find could be $O(n)$
   - Union $O(1)$

2. For every element, we can keep the set identification

Array



Find is $O(1)$ array look up

Union : $O(n)$: change the labels of the elements involved

When we do union, let us change the labels of the elements in the smaller set

The cost of a sequence of at most $n-1$ union operations can be charged to the number of label changes over all elements

Let $n(x)$ be the no. of label changes incurred by x over the entire sequence of $n-1$ unions.

Then cost of $n-1$ unions $= \sum_x n(x)$

Claim $n(x) \leq \log_2 n$

For every label change x is in a set which is at least twice the size of the previous set.

So total cost of unions $\leq n \cdot \log n$

Total cost of m Finds and $n-1$ unions is bounded by $O(m + n \log n)$