GREEDY TECHNIQUE
Tuesday, October 16, 2018 5:10 PM

- brute / exhaustive search
- transform
- decrease
- divide

* GREEDY : - method of choice for optimization problems
  - works great when it applies, but must provide a proof

Ex: COIN CHANGE

item: \$ 2.56

pay: \$ 3.00

change: \$0.44 : Q + D + N + P + P + P + P

25 10 5 1 1 1 1

This solution MINIMIZES the number of coins (7 coins)

D + D + D + D + P + P + P + P   8 coins

P + . - — - — - P ?   44 coins

HOW IS THE OPTIMAL SOLUTION CONSTRUCTED ?

- coin by coin
- each time pick the largest possible denomination (less or equal to remaining amount)

$\boxed{n = 44}$

P N D Q

1 5 10 25

× ×

Q D N P P P P

$44 = Q + 19$

n = 19

n = 9

n = 4

3

2

1

0

greedy-cc(n)

{

if (n==0)

```
        return {};
    D[] = {1,5,10,25};
    for (i = 3; D[i] > n; --i)
        ;
    return {D[i]} ∪ greedy-cc(n-D[i]);
}
```

Note: 1) this is a DECREASE-CONQUER alg
2) in each step, a "local" optimal choice is made among the options
these local optimal choices happen to produce an global optimal solution.

CLAIM: greedy-cc(n) returns the (unique) optimal solution.

Proof: Induction on n

Base case: $n = 0$. Clearly {} is the optimal solution.

Induction: when $n > 0$, we claim that the optimal solution MUST contain a coin of largest denomination $\le n$.

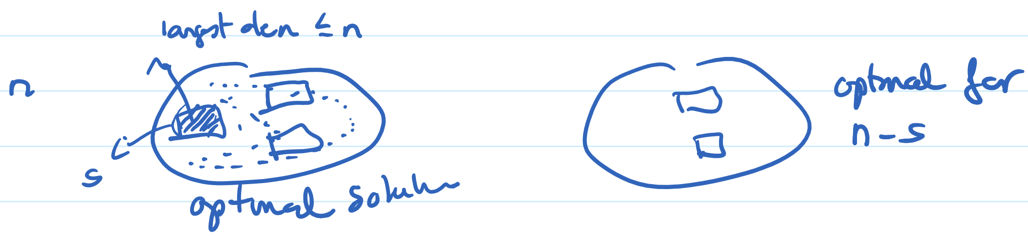
~~Case a~~: $1 \le n \le 4$: optimal solution consists of pennies only
so it contains a coin of largest denomination $\le n$.

Case b: $5 \le n \le 9$: largest den $\le n$ is nickel (5)
Suppose not. Then any optimal solution must consist of pennies only and has at most 4 pennies. But this means $n \le 4$, contradiction

Case c: $10 \le n \le 24$: largest den $\le n$ is dime (10)
Suppose not. Then any optimal solution must consist of pennies and nickels. At most 4 pennies + 1 nickel = 9

Case d: $n \geq 25$: largest den $\le n$ is quarter (25)
Suppose not. Then any optimal solution consists of

pennies,	nickels,	dimes only		
4	0	2	= 4+20 = 24	$\} < 25$
4	1	1	= 4+5+10 = 19	



let O be an optimal solution for n, and let s be largest den $\le n$

O - {s} must be an optimal solution for n - s [OPTIMAL SUBSTRUCTURE property]

By induction hyp, greedy-cc(n - D(i)) returns an optimal solution for n - D(i)
so greedy-cc returns an optimal solution for n.

---

coin change problem for denominations 1, 10, 25
Does the greedy strategy still work? [always pick largest possible deno]

Counter-example:

n = 30

greedy: Q + P + P + P + P + P   optimal: D + D + D

n = 5

coin change: denomination 1, 5, 7, 10, 25

Counter-example:

n = 14

greedy: D + P + P + P + P   optimal: 7 + 7

SCHEDULING PROBLEM:

INPUT: n print jobs; job i has length l_i

OUTPUT: an ordering of the jobs to minimize the total wait time.

wait time = sum of job lengths of those jobs ahead of this one

Ex: 4, 1, 7, 3

	WAIT		
4	0	1	0
1	4	3	1
7	5	4	4
3	12	7	8
	21		13

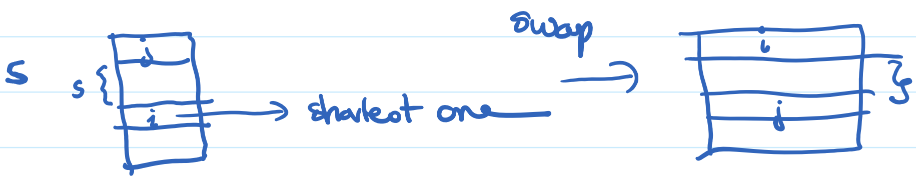
```
print_job ( l[lo..hi] )
{ if (lo ≤ hi)
  {
    build_heap (l);
    return {l[1]} + print_job (l[2..n])
  }
}
```

```
{ build_heap(l);
  return {l[1]} + print_job(l[2..n])
}
}
```

CLAIM: The greedy solution provides an optimal solution for the PAINT JOB problem

Proof:

Let O be an optimal solution. The first scheduled job must be the shortest. If not, swap it with the shortest one.



$-(j-i) - (S+j) + S+i$

Total change: $-(j-i)$ for each job between i, j

$-\cancel{S}-j$ for i

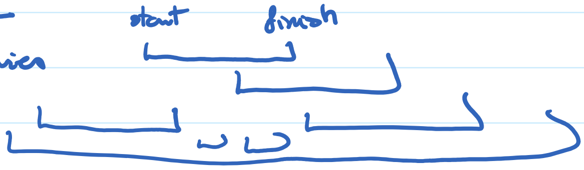
$+\cancel{S}+i$ for j

≤ 0 because $j > i$

Further, the remaining ordering must be optimal for all the jobs minus the shortest one.

MOVIE PROBLEM

Input: n movies



Output:

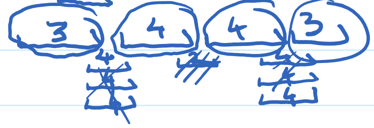
largest collection of nonoverlapping movies.

Multiple greedy strategies:

earliest first: counter example

shortest first: "

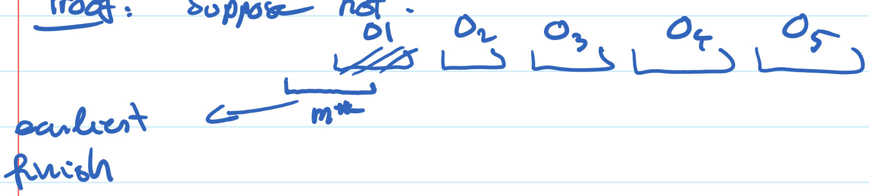
fewest overlap:



FINISH FIRST

CLAIM: There is **AN** OPTIMAL SOLUTION that includes the movies with earliest finish time

Proof: Suppose not.



= optimal solution

Let O be any optimal solution. If O contains m^* (the earliest finish time) then we are done.

If not, let O_1 be the first movie that finishes in O.

$O - \{o_1\} \cup \{m^*\}$ is another optimal solution, because m^* does not overlap with the other movies in O.

```
movie ( M[lo..hi] )
{
    if ( lo ≤ hi )
    {
        m* = movie with least finish time
        swap m* with M[lo]
        return {m*} ∪ movie ( M[lo+1..hi] ) ;
    }
}
```

FANO - SHANNON CODE