

Introduction to Algorithms

Greedy Algorithms

Greedy Algorithms



Coin Changing Problem
Greedy Algorithm

Greedy Strategy

Goal: Given currency denominations: 1, 5, 10, 25, 100, give change to customer using *fewest* number of coins.

Ex: 34¢.



Cashier's algorithm: At each iteration, give the *largest* coin valued \leq the amount to be paid.

Ex: \$2.89.



Greedy is not always Optimal

Observation: Greedy algorithm is sub-optimal for US postal denominations: 1, 10, 21, 34, 70, 100, 350, 1225, 1500.

Counterexample. 140¢.

Greedy: 100, 34, 1, 1, 1, 1, 1, 1.

Optimal: 70, 70.



Lesson: Greedy is short-sighted. Always chooses the most attractive choice at the moment. But this may lead to a dead-end later.

Greedy Algorithms Outline

Pros

- Intuitive
- Often simple to design (and to implement)
- Often fast

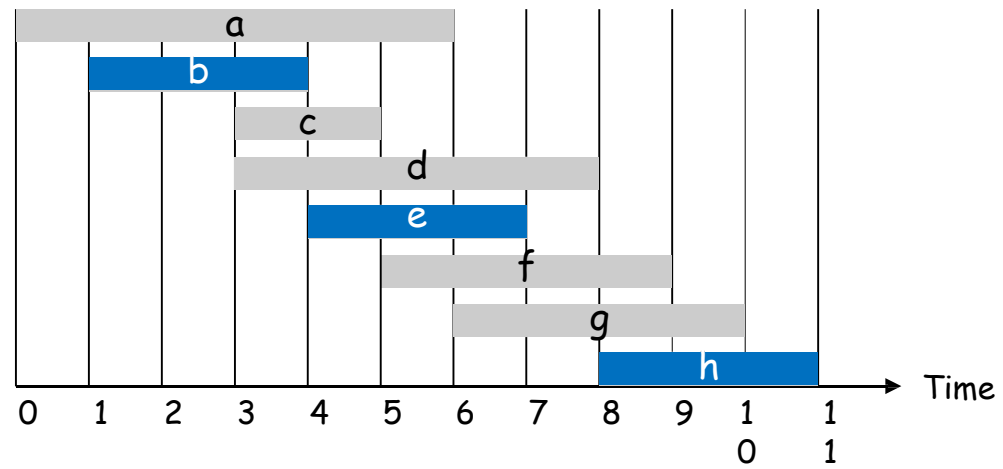
Cons

- Often incorrect!

Proof techniques:

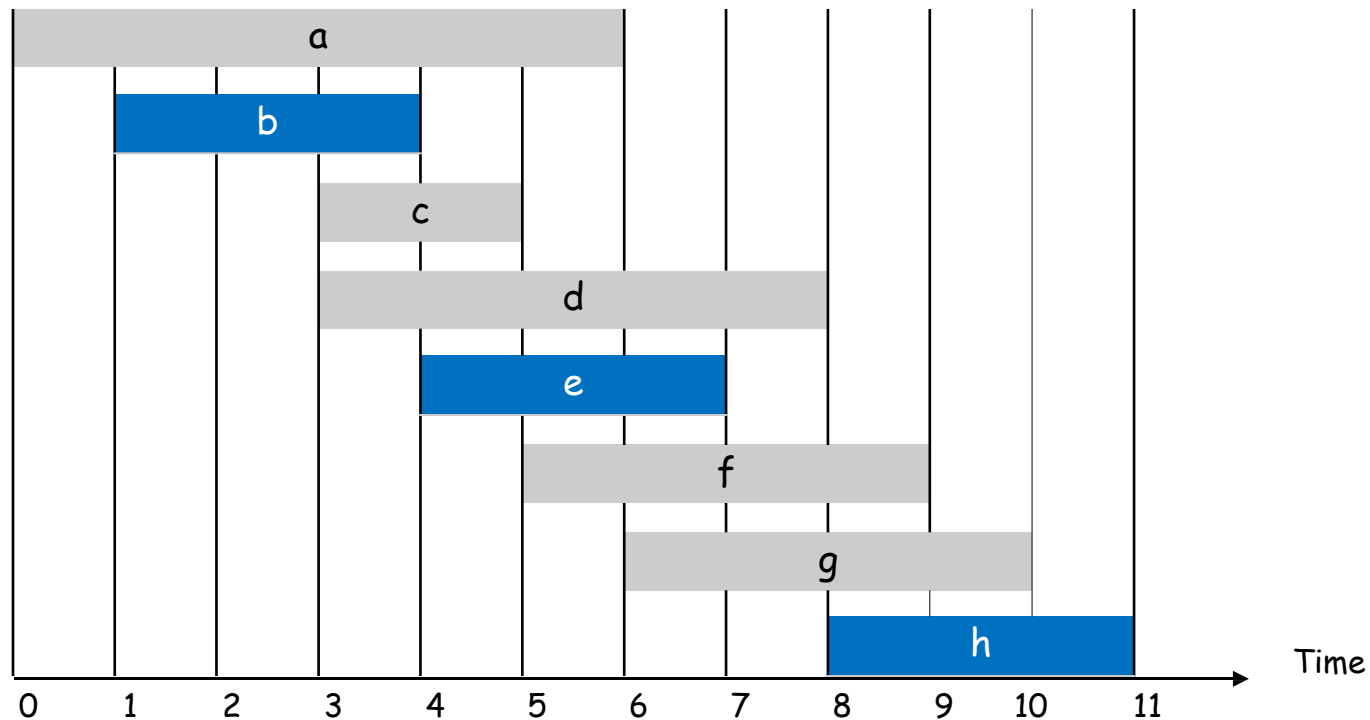
- Stay ahead
- Structural
- Exchange arguments

Interval Scheduling



Interval Scheduling

- Job j starts at $s(j)$ and finishes at $f(j)$.
- Two jobs **compatible** if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.



Greedy Strategy

Sort the jobs in **some** order. Go over the jobs and take as much as possible provided it is compatible with the jobs already taken.

Main question:

- What order?
- Does it give the Optimum answer?
- Why?

Possible Approaches for Inter Sched

Sort the jobs in **some** order. Go over the jobs and take as much as possible provided it is compatible with the jobs already taken.

[Earliest start time] Consider jobs in ascending order of start time s_j .

[Earliest finish time] Consider jobs in ascending order of finish time f_j .

[Shortest interval] Consider jobs in ascending order of interval length $f_j - s_j$.

[Fewest conflicts] For each job, count the number of conflicting jobs c_j . Schedule in ascending order of conflicts c_j .

Greedy Alg: Earliest Finish Time

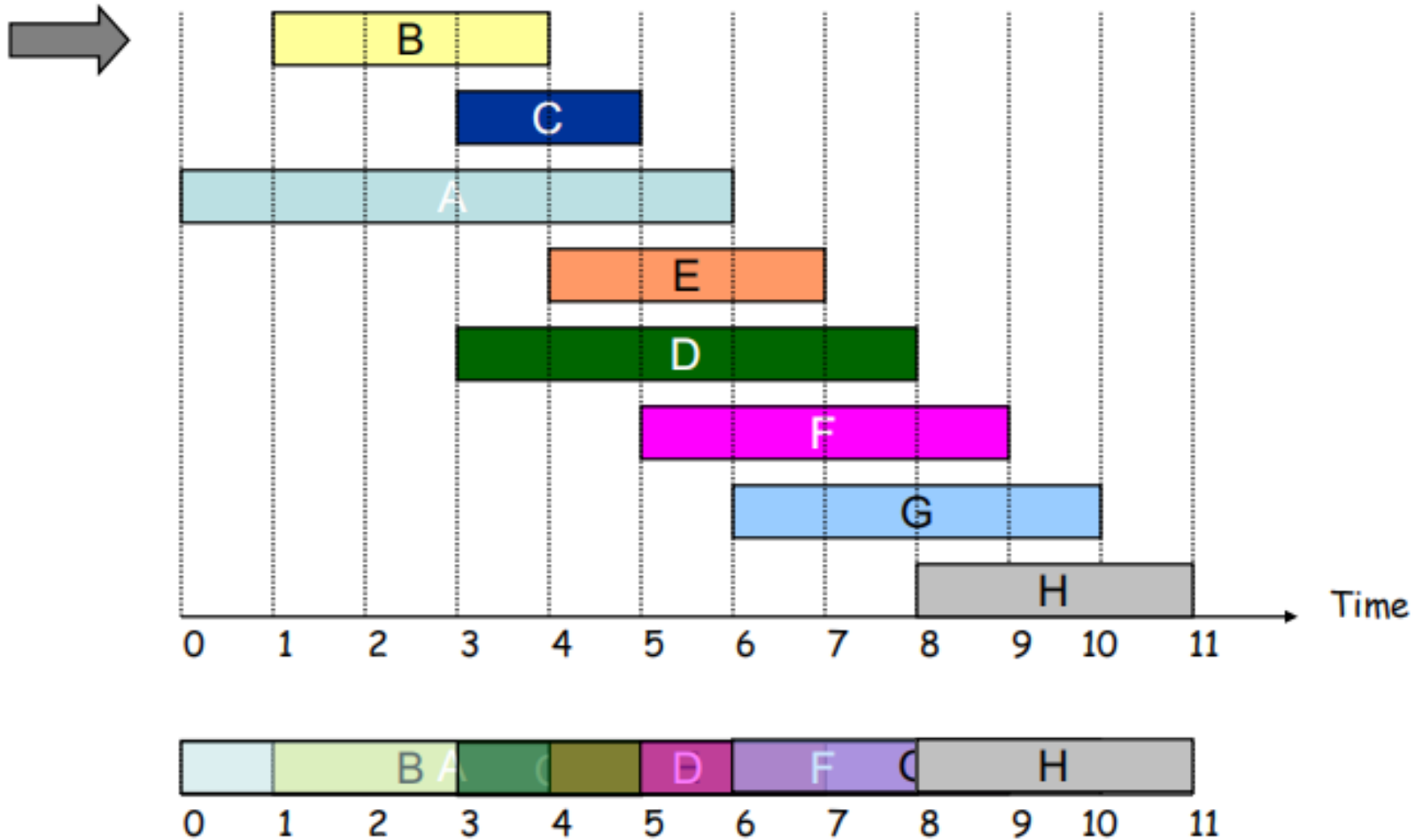
Consider jobs in increasing order of finish time. Take each job provided it's compatible with the ones already taken.

```
Sort jobs by finish times so that  $f(1) \leq f(2) \leq \dots \leq f(n)$ .  
 $A \leftarrow \emptyset$   
for  $j = 1$  to  $n$  {  
    if (job  $j$  compatible with  $A$ )  
         $A \leftarrow A \cup \{j\}$ .  
}  
return  $A$ 
```

Implementation. $O(n \log n)$.

- Remember job j^* that was added last to A .
- Job j is compatible with A if $s(j) \geq f(j^*)$

Greedy Alg: Example



Correctness

Theorem: Greedy algorithm is optimal.

Pf: (technique: “Greedy stays ahead”)

Let i_1, i_2, \dots, i_k be jobs picked by greedy, j_1, j_2, \dots, j_m those in some optimal solution in order.

We show $f(i_r) \leq f(j_r)$ for all r , by induction on r .

Base Case: i_1 chosen to have min finish time, so $f(i_1) \leq f(j_1)$.

IH: $f(i_r) \leq f(j_r)$ for some r

IS: Since $f(i_r) \leq f(j_r) \leq s(j_{r+1})$, j_{r+1} is among the candidates considered by greedy when it picked i_{r+1} , & it picks min finish, so $f(i_{r+1}) \leq f(j_{r+1})$

Observe that we must have $k \geq m$, else j_{k+1} is among (nonempty) set of candidates for i_{k+1}