

greedy method

# activity selection problem

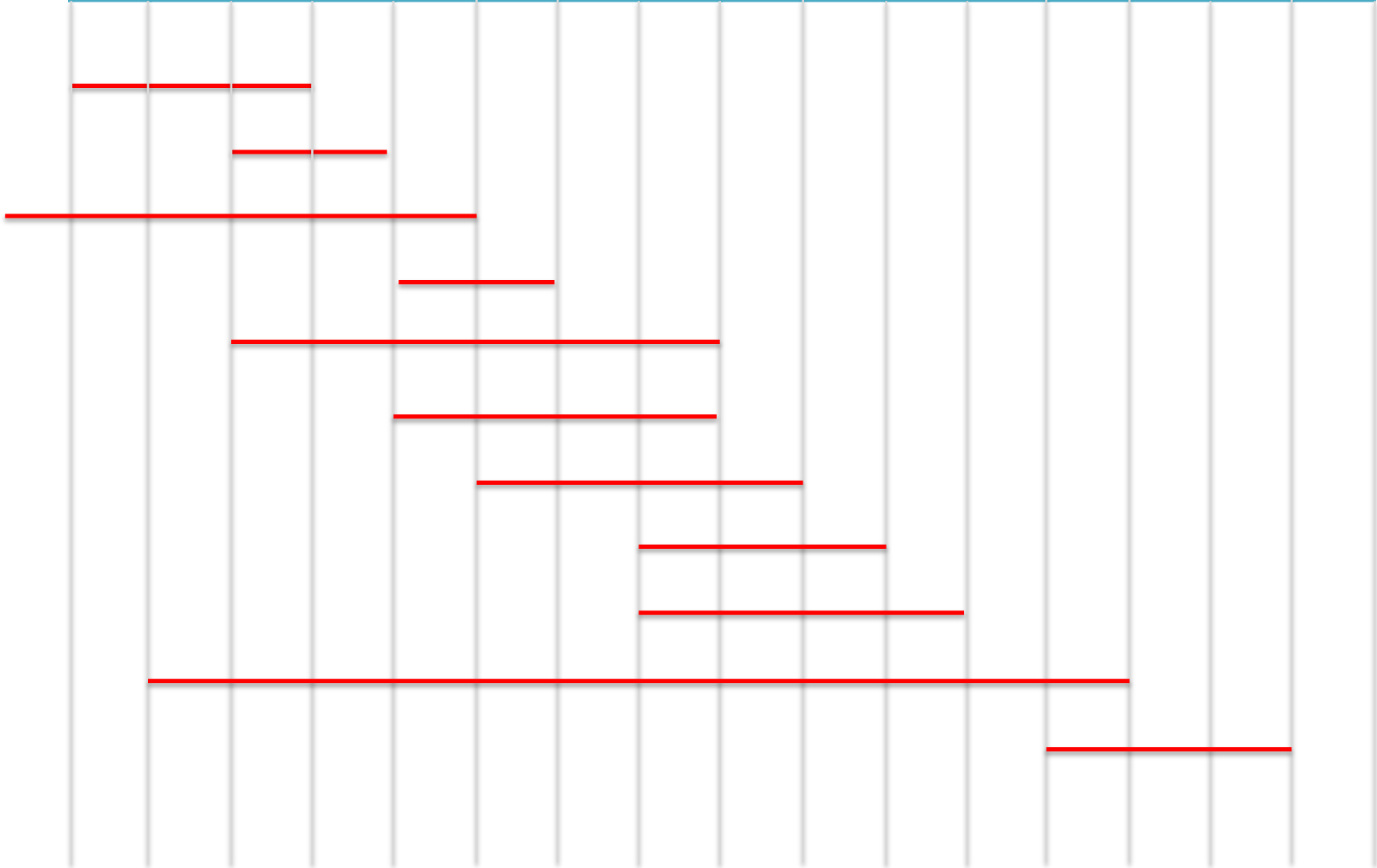
- given set  $S = \{a_1, a_2, \dots, a_n\}$  of possible activities
- each  $a_i$  has a **start** time  $s_i$  and **finish** time  $f_i$
- $0 \leq s_i < f_i < \infty$
- $a_i$  takes place during  $[s_i, f_i)$  (**half-open interval**)
- activities  $a_i$  and  $a_j$  are **compatible** if their intervals  $[s_i, f_i)$  and  $[s_j, f_j)$  do not overlap
- **problem**: select maximum size subset of compatible activities

# example

i	1	2	3	4	5	6	7	8	9	10	11
s	1	3	0	5	3	5	6	8	8	2	12
f	4	5	6	7	9	9	10	11	12	14	16

- $\{a_3, a_9, a_{11}\}$  is a mutually compatible set
- but is not maximum
- both  $\{a_1, a_4, a_8, a_{11}\}$  and  $\{a_2, a_4, a_9, a_{11}\}$  are larger
- this is sorted by finish time (useful for greedy method, but we're not there yet)

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
---	---	---	---	---	---	---	---	---	----	----	----	----	----	----	----



# dynamic programming approach

define  $S_{ij} = \{ a_k \mid f_i \leq s_k \text{ and } f_k \leq s_j \}$

the set of activities that start after  $a_i$  finishes and finish before  $a_j$  starts – that is, they can be scheduled between  $a_i$  and  $a_j$

**subproblem:** let  $c[i,j]$  be the size of the largest solution for  $S_{ij}$

try scheduling all possible  $k$

recurrence

- $c[i,j] = 0$  if  $S_{ij} = \emptyset$
- otherwise  $c[i,j] = \text{MAX} \{ c[i,k] + c[k,j] + 1 \mid a_k \in S_{ij} \}$

# greedy choice

always pick the earliest finishing activity

define  $S_k = \{ a_i \mid f_k \leq s_i \}$

- activities starting after  $a_k$  is done

**theorem 16.1 (p 418)** Consider any non-empty subproblem  $S_k$ , and let  $a_m$  be an activity in  $S_k$  with the earliest finish time. Then  $a_m$  is included in some maximum-size subset of mutually compatible activities of  $S_k$ .

proof in text

# greedy algorithms need

- subproblems and
- optimal substructure, just like dynamic programming
- a greedy choice theorem (**important**)

# recursive algorithm

RAS (Recursive Activity Selector) on inputs  $s$  (start times),  $f$  (finish times),  $k$  (subproblem), and  $n$

```
RAS( $s, f, k, n$ )  
   $m = k + 1$   
  while  $m \leq n$  and  $s[m] < f[k]$   
     $m = m + 1$   
  if  $m \leq n$   
    return  $\{a_m\} \cup \text{RAS}(s, f, m, n)$   
  else return  $\emptyset$ 
```

find first  $a_m$  in  $S_k$   
to finish

finish times already sorted,  
time now looks like  $O(n)$



# iterative version

```
A={a1}  
k=1  
for m=2 to n  
    if s[m] ≥ f[k]  
        A=A ∪ {am}  
        k=m  
return A
```

time is pretty clearly  $O(n)$ , but don't forget the  $O(n/\lg n)$  time used to sort  $f$

# other similar problems

- (ex 16.1-4) interval coloring: given activities as above, assign the fewest number of colors to each activity so that no two overlapping activities have the same color
- (ex 16.1-5) same as activity selection, but in addition give each activity  $a_i$  a value  $v_i$ , and maximize the total value of selected activities

# greedy strategy (§16.2)

1. determine optimal substructure of the problem
2. develop a recursive solution
3. show that if greedy choice made, only one subproblem remains
4. prove it is always safe to make greedy choice
5. develop recursive algorithm implementing greedy
6. convert recursive version to iterative

# matroids – a theory of greed

A **matroid** is an ordered pair  $M=(S,I)$  satisfying:

- 1)  $S$  is a finite set
- 1)  $I$  is a family of subsets of  $S$ , called the **independent** sets. It must have the **hereditary** property: if  $B \in I$  and  $A \subseteq B$ , then  $A \in I$
- 1)  $M$  has the **exchange** property: if  $A \in I$ ,  $B \in I$ , and  $|A| < |B|$ , then there is an  $x \in B - A$  such that  $A \cup \{x\} \in I$

# applied to graphs

Given a graph  $G=(V,E)$ , the graphic matroid of  $G$  is the structure  $M_G=(S_G,I_G)$  where

- $S_G$  is the set of edges  $E$
- If  $A$  is a set of edges, then  $A \in I_G$  iff  $A$  is acyclic. That is, the independent sets of edges of  $G$  are those that form a forest.

we can also add a weight function  $w:S \rightarrow \mathbb{N}$  to any matroid

# generic greedy algorithm on matroid

```
greedy(M, w)
  A = ∅
  sort the elements S of M by weight w
  for each x ∈ S taken in order
    if A ∪ {x} ∈ I
      then A = A ∪ {x}
  return A
```

looks like MST

also, the dual of a matroid is a matroid  
for graphs, the dual I are sets of edges containing a cycle  
the combinatorics gets complicated