Greedy Algorithms

- Given a set of activities (s_i, f_i), we want to schedule the maximum number of non-overlapping activities.
- GREEDY-ACTIVITY-SELECTOR (s, f)

```
n = length[s]
S = {a_1}
i = 1
for m = 2 to n do
        if s_m is not before f_i then
                S = S U {a_m}
                i = m
return S
```

- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14] -- Sorted by finish times
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]
- [1,4], [3,5], [0,6], [5,7], [3,8], [5,9], [6,10], [8,11], [8,12], [2,13], [12,14]

Why does it work?

- THEOREM

  Let A be a set of activities and let a_1 be the activity with the earliest finish time. Then activity a_1 is in some maximum-sized subset of non-overlapping activities.

- PROOF

  Let S' be a solution that does not contain a_1. Let a'_1 be the activity with the earliest finish time in S'. Then replacing a'_1 by a_1 gives a solution S of the same size.

  Why are we allowed to replace? Why is it of the same size?

Greedy Algorithms – Huffman Coding

- Huffman Coding Problem

  Example: Release 40.42 of 31-Jan-2003 of Swiss-Prot Protein Database contains 121,745 sequence entries, comprising 44,680,829 amino acids. There are 20 possible amino acids. What is the minimum number of bits to store the compressed database?

  ~250 M bits or 30MB.

- How to improve this?
- Information: Frequencies are not the same.

Ala (A) 7.72	Gln (Q) 3.91	Leu (L) 9.56	Ser (S) 6.98
Arg (R) 5.24	Glu (E) 6.54	Lys (K) 5.96	Thr (T) 5.52
Asn (N) 4.28	Gly (G) 6.90	Met (M) 2.36	Trp (W) 1.18
Asp (D) 5.28	His (H) 2.26	Phe (F) 4.06	Tyr (Y) 3.13
Cys (C) 1.60	Ile (I) 5.88	Pro (P) 4.87	Val (V) 6.66

Huffman Coding

- **Idea:** Use shorter codes for more frequent amino acids and longer codes for less frequent ones.

Greedy Algorithms – Other examples

- Minimum Spanning Trees (Kruskal's & Prim's)
- Matroid Problems
- Several scheduling problems

Dynamic Programming

- Activity Problem Revisited: Given a set of activities (s_i, f_i), we want to schedule the maximum number of non-overlapping activities.
- New Approach:

  A_i = Best solution for intervals $\{a_1, \ldots, a_i\}$ that includes interval a_i

  B_i = Best solution for intervals $\{a_1, \ldots, a_i\}$ that does not include interval a_i
- Does it solve the problem to compute A_i and B_i?
- How to compute A_i and B_i?