We’ve done

- Fast Fourier Transform
  - Polynomial Multiplication

Now

- Introduction to the greedy method
  - Activity selection problem
  - How to prove that a greedy algorithm works
  - Huffman coding

Next

- Matroid theory
Greedy Algorithms

- The second algorithm design technique we learn
- Used to deal with optimization problems
- Optimization problems: find an optimal solution among a large set of candidate solutions
  - 0-1 **knapsack problem**: A robber found \( n \) items in a store, the \( i \)th item is worth \( v_i \) dollars and weighs \( w_i \) pounds (\( v_i, w_i \in \mathbb{Z}^+ \)), he can only carry \( W \) pounds. Which items should he take?
  - **Traveling Salesman Problem (TSP)**: find the shortest route for a salesman to visit each of the \( n \) given cities once, and return to the starting city.
- Different than brute-force
- Characterized by
  - Greedy-choice property
  - Optimal substructure
The Activity-Selection Problem

• Has to do with scheduling of resources (class room, CPU)

• Input:
  – a set of activities $A = \{a_1, \ldots, a_n\}$ to be scheduled
  – activity $a_i$ spends the time interval $[s_i, f_i)$

• Output: a set of as many activities as possible with no time conflict
A Greedy Algorithm

Arrays $S$ and $F$ store start and finish times:

$$S[i] = s_i, \quad F[i] = f_i.$$

Activity-Selection($S, F, n$)

1: Sort $F$ in increasing order
2: Simultaneously rearrange $S$ correspondingly
3: $C \leftarrow \{1\}$ // pick the first activity
4: $j \leftarrow 1$ // record the last chosen activity
5: for $i \leftarrow 2$ to $n$ do
6: \hspace{1em} if $s_i \geq f_j$ then
7: \hspace{2em} $C \leftarrow C \cup \{i\}$ // add $i$ to the output set
8: \hspace{2em} $j \leftarrow i$ // record the last chosen activity
9: \hspace{1em} end if
10: end for
11: Output $C$
Why does it work?

- Remember the objective: maximize the number of scheduled activities

- **Want:** show that the algorithm’s output is optimal

- **Greedy-choice property:** At every step there exists an optimal solution which contains the greedy choice (the first interval)
  - This shows that we are on the right track to get to **an** optimal solution

- **Optimal substructure:** Are we still on the right track at the next step?
  - At the next step: we try to solve the same problem with the set \( A' \) of activities compatible with the first choice
  - If \( O \) is an optimal solution to the original problem containing \( \{1\} \), then \( O' = O - \{1\} \) is an optimal solution to \( A' \)
Elements of the Greedy Strategy

- Question: in the Activity Selection Problem, might there be an optimal solution which does not contain the greedy choice?

- At every step, the choice we made narrows down the search

- Make sure we do not narrow it down to zero

- **Greedy-choice property**: There exists an optimal solution which contains the greedy choice

- **Optimal substructure**:
  - An optimal solution to the problem contains within it an optimal solution to the subproblem
  - After each greedy choice is made, we are left with an optimization problem of the same form as the original problem
Knapsack Problems

0-1 knapsack problem

- **Input**: $n$ items, the $i$th item has value $v_i$ dollars and weighs $w_i$. A maximum weight $W$. $v_i, w_i, W \in \mathbb{Z}^+$.  
- **Output**: a set of items as valuable as possible with total weight at most $W$.

Fractional knapsack problem

- **Input**: $n$ items, the $i$th item has value $v_i$ dollars and weighs $w_i$. A maximum weight $W$. $v_i, w_i, W \in \mathbb{Z}^+$.  
- **Output**: a set of items as valuable as possible with total weight at most $W$.  
- **Relaxation**: can take any fraction of an item.
Huffman Codes

- 7-bit ASCII code for “abbccc” uses 42 bits
- Suppose we use ’0’ to code ’c’, ’10’ to code ’b’, and ’11’ to code ’c’: “111010000” - 9 bits
- To code effectively:
  - Variable codes
  - No code of a character is a prefix of a code for another: prefix code
  - The characters with higher frequencies should get shorter codes
- Prefix codes can be represented by binary trees with characters at leaves
- The binary trees have to be full if we want the code to be optimal (why?)
- The problem: given the frequencies, find an optimal full binary tree
Huffman’s Greedy Algorithm

- **Input:**
  - $C$: the set of characters
  - Frequency $f(c)$ for each $c \in C$

- **Output:** an optimal coding tree $T$.
  
  Let $d_T(c)$ be the depth of a leaf $c$ of $T$
  
  The total number of bits required is

  $$B(T) = \sum_{c \in C} f(c)d_T(c)$$

  We want to find $T$ with the least $B(T)$

Huffman’s Idea

1. **while** there are two or more leaves in $C$ **do**
2. Pick two leaves $x, y$ with least frequency
3. Create a node $z$ with two children $x, y$, and frequency
   $$f(z) = f(x) + f(y)$$
4. $C = (C - \{x, y\}) \cup \{z\}$
5. **end while**
Correctness of Huffman Coding

Greedy-Choice Property

**Lemma 1.** Let $C$ be a character set, where each $c \in C$ has frequency $f(c)$. Let $x$ and $y$ be two characters with least frequencies. Then, there exists an optimal prefix code for $C$ in which the codewords for $x$ and $y$ have the same length and differ only in the last bit.

Optimal Substructure

**Lemma 2.** Let $T$ be a full binary tree representing an optimal prefix code for $C$. Let $x$ and $y$ be any leaves of $T$ which share the same parent $z$. Let $C' = (C - \{x, y\}) \cup \{z\}$, with $f(z) = f(x) + f(y)$. Then, $T' = T - \{x, y\}$ is an optimal tree for $C'$. 
Things to remember

- To prove greedy choice property:
  - Show that there exists an optimal solution which “contains” the greedy choice
  - A common method: take any optimal solution $O$, try modifying $O$ to $O'$, so that $O'$ is still optimal, and $O'$ contains the greedy choice

- To prove optimal substructure:
  - Let $O_1$ be an optimal solution which contains the greedy choice. Show that $O_1$ minus the greedy choice (resulting in $O'_1$, say) is an optimal solution to the subproblem.
  - A common method: assume $O'_1$ is not optimal for the subproblem, then there is some optimal solution $O'_2$ of the subproblem. Then, construct a solution $O_2$ of the original problem from $O'_2$ and the greedy choice, such that $O_2$ is a better solution than $O_1$. Contradiction!