http://www.cs.unh.edu/~ruml/cs758

Greedy Algorithms

Greedy

Make best *local* choice, then solve remaining subproblem.

Eg, optimal solution uses the greedy choice + optimal solution to remaining subproblem.

Unlike DP, haven't already solved subproblems, don't need to pick 'best' subsolution to use.

Activity Selection

Given n activities, $\{1, 2, ..., n\}$; the ith activity corresponding to an interval starting at $s(i)$ and finishing at $f(i)$, find a compatible set with maximum size.

Activity Selection

Given n activities, $\{1, 2, ..., n\}$; the ith activity corresponding to an interval starting at $s(i)$ and finishing at $f(i)$, find a compatible set with maximum size.

Make a choice: at each step, select the next activity to include in the set.

Is there a rule?

"Rules" for Activity Selection

- Earliest start time
- Earliest finish time
- Smallest interval
- Least conflicts

Try to make a decision that is good locally,
before solving remaining subproblem.

Is best decision independent of remaining solution?

“Rules” for Activity Selection

- Earliest start time
- Earliest finish time
- Smallest interval
- Least conflicts

Try to make a decision that is good locally,
before solving remaining subproblem.

Is best decision independent of remaining solution?

The Algorithm

Make greedy choice, then solve remaining subproblem:

1. $R \leftarrow$ all activities
2. $A \leftarrow \{\}$
3. while $R \neq \{\}$
4. let $b =$ activity in R with earliest finish time
5. $R \leftarrow R \setminus \{c : c$ conflicts with $b\}$
6. $A \leftarrow A \cup \{b\}$
7. return A

Is this optimal?

Proving Greedy Optimal

Need to show:

1. greedy choice is optimal: there exists an optimal solution that uses it
2. optimal substructure: the remaining subproblem can be solved the same way

The Greedy Choice Property

Prove that first choice in optimal solution can be made greedily:

- Let $\langle a_1, a_2, ..., a_i \rangle$ be an optimal schedule.
- If a_1 is the activity with the earliest finish time then the greedy choice is within some optimal solution.
- If a_1 is not the greedy choice (activity with the earliest finish time) then there must exist an activity b with an earlier finish time ($f(b) < f(a_1)$).
- b (the greedy choice) will be compatible with a_2, so $\langle b, a_2, ..., a_i \rangle$ is also an optimal solution.

So making the greedy choice is always compatible with an optimal solution.

Optimal Substructure

Prove that optimal solution contains optimal solution to remaining subproblem after greedy choice:

- Let $\langle a_1, a_2, ..., a_i \rangle$ be an optimal schedule.
- For the sake of contradiction, assume $\langle a_k, ..., a_i \rangle$ is a suboptimal sub-schedule for the time after activity a_{k-1}.
- So, there exists a sequence $\langle b_1, ..., b_j \rangle$ that is a better schedule for this time interval (after $f(a_{k-1})$).
- 'cut and paste': replacing the suboptimal sub-schedule with the better one yields a feasible schedule.
- Since the new sub-schedule includes more activities, $\langle a_1, ..., a_{k-1}, b_1, ..., b_j \rangle$ must be a better schedule.
- But this implies our optimal schedule was suboptimal: contradiction!
- So our assumption must not hold. Every sub-sechedule must be optimal for its interval.

Summary of Greedy Algorithms

Make best *local* choice, then solve remaining subproblem.

Eg, optimal solution uses the greedy choice + optimal solution to remaining subproblem.

1. prove greedy choice is safe (an optimal solution uses that choice): subsitute greedy choice in optimal soluion
2. prove optimal substructure (optimal solution uses optimal solutions of subproblems): assume suboptimal, then derive contradiction

Break

- asst7
- midterm Thu Oct 17
- midterm review Fri Oct 11

Huffman Coding

The Problem

Given a table of character frequencies, find a set of prefix-free codewords that minimizes encoding length:

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

c	$f(c)$	code
a	5	1
b	2	00
c	1	01

a a a b a b a c $\Rightarrow$ 1 1 1 00 1 00 1 01

regular ASCII: 8 bytes = 64 bits $\Rightarrow$ 11 bits ($\sim$83% smaller)
fixed size: $8\times$ 2 bits = 16 bits $\Rightarrow$ 11 bits ($\sim$31% smaller)

Code Structure

frequent characters will have shorter codes

every node in the optimal code tree has two children

The Algorithm

Distinguish elements by penalizing the two least frequent:

1. $C \leftarrow$ characters c tagged by frequency $f(c)$
2. $Q \leftarrow$ MAKE-MIN-HEAP(C)
3. for $i = 1$ to $|C| - 1$ do
4. let z be a new tree node
5. $z.left \leftarrow$ EXTRACT-MIN(Q)
6. $z.right \leftarrow$ EXTRACT-MIN(Q)
7. $f(z) \leftarrow f(z.left) + f(z.right)$
8. HEAP-INSERT(Q, z)
9. return EXTRACT-MIN(Q)

What's the worst-case time complexity?

Proving that Greedy is Optimal

Show that

1. greedy choice is optimal (optimal solution can use greedy choice)
2. the greedy choice plus an optimal solution to the remaining subproblem is an optimal solution for the larger problem

The Greedy Choice is Optimal

Any code not worse (and maybe better) with greedy choice:

Let x and y be the least frequent and a and b be siblings at the deepest depth in T. If they are not the same, we can improve the code by swapping x and y for a and b.

Proof: Consider swapping x and a to get T'.

$$
\begin{aligned}
B(T) - B(T') &= \sum_{c \in C} f(c) \cdot d_T(c) - \sum_{c \in C} f(c) \cdot d_{T'}(c) \\
&= f(a) \cdot d_T(a) + f(x) \cdot d_T(x) \\
&\quad - f(a) \cdot d_{T'}(a) - f(x) \cdot d_{T'}(x) \\
&= f(a) \cdot d_T(a) + f(x) \cdot d_T(x) \\
&\quad - f(a) \cdot d_T(x) - f(x) \cdot d_T(a) \\
&= (f(a) - f(x))(d_T(a) - d_T(x)) \\
&\geq 0
\end{aligned}
$$

Optimal Substructure

Show that the optimal solution to the subproblem remaining after the greedy choice has been made can be extended by the greedy choice into the optimal solution.

Combine least frequent characters x and y in C into z with $f(z) = f(x) + f(y)$. Let T_R be the optimal code tree for this reduced set C_R. Now expand leaf for z in T_R into branch for leaves x and y. Prove this expanded tree T is optimal for C.

Optimal Substructure Proof, Part 1/2

Combine least frequent characters x and y in C into z with $f(z) = f(x) + f(y)$. Let T_R be the optimal code tree for this reduced set C_R. Now expand leaf for z in T_R into branch for leaves x and y. Prove this expanded tree T is optimal for C.

First, compare encoding costs where T and T_R differ:

$$\begin{aligned} f(x) \cdot d_T(x) + f(y) \cdot d_T(y) &= (f(x) + f(y))(d_{T_R}(z) + 1) \\ &= f(z) \cdot d_{T_R}(z) + (f(x) + f(y)) \end{aligned}$$

Rest of the trees are the same, so:

$$\begin{aligned} B(T) &= B(T_R) + f(x) + f(y) \\ B(T_R) &= B(T) - f(x) - f(y) \end{aligned}$$

Optimal Substructure Proof, Part 2/2

Combine least frequent characters x and y in C into z with $f(z) = f(x) + f(y)$. Let T_R be the optimal code tree for this reduced set C_R. Now expand leaf for z in T_R into branch for leaves x and y. Prove this expanded tree T is optimal for C.

We just showed $B(T_R) = B(T) - f(x) - f(y)$.

Now, assume T non-optimal for C but tree O is. Note x and y are siblings in O by greedy choice property. Form O_R by replacing them with z. Encoding cost:

$$\begin{aligned} B(O_R) &= B(O) - f(x) - f(y) \text{ by prev argument} \\ &< B(T) - f(x) - f(y) \text{ by assumption about } O \\ &< B(T_R) \end{aligned}$$

But T_R was optimal for C_R — contradiction!
Suboptimal T is impossible with optimal T_R.

Summary of Greedy Algorithms

Make best *local* choice, then solve remaining subproblem.

Eg, optimal solution uses the greedy choice + optimal solution to remaining subproblem.

1. prove greedy choice is safe (an optimal solution uses that choice): subsitute greedy choice in optimal soluion
2. prove optimal substructure (optimal solution uses optimal solutions of subproblems): assume suboptimal, then derive contradiction

EOLQs

For example:

- What's still confusing?
- What question didn't you get to ask today?
- What would you like to hear more about?

Please write down your most pressing question about algorithms and put it in the box on your way out.
Thanks!