last time: elementary graph algorithms

today: greedy algorithms

A greedy algorithm chooses the locally optimal choice at each stage with the hope of finding the global optimum. It lists all the possible local ways and then cuts some branches repeatedly to arrive at the final solutions.

Activity-selection problem

Given: n activities $a_1, a_2, ..., a_n$, each activity has start time $s_i$, finish time $f_i(0 \leq s_i < f_i < \infty)$. The activities compete for resource and cannot be overlapped.

Goal: find a maximum number of mutually compatible activities that can be scheduled.

From above, we know that some activities can be compatible, some cannot. The shorter one is better to be compatible.
The solutions are obvious: \{10\}, \{5, 9, 11\}, \{1, 4, 9, 11\}, ...

How to solve this problem? Is the solution optimal?

Arrange activities according to ascending finish time and introduce 2 artificial activities: \(f_0 = 0, s_{n+1} = \infty\).

\(S_{i,j}\) --- the set of activities compatible with \(a_i\) and \(a_j\). The finish time of \(a_i\) is before start time of \(a_j\).

\(A_{i,j}\) --- the optimal solution for elements in \(S_{i,j}\).

\(c_{i,j}\) --- the maximum number of mutually compatible activities in \(S_{i,j}\). \(c_{i,j} = |A_{i,j}|\)

\[
\begin{array}{c}
\text{find } S_{i,j} \\
\hline
\text{a}_i \\
\text{find } S_{0,i} \\
\hline
\text{find } S_{k,n+1}
\end{array}
\]

For example, \(c_{2,11} = 3\), \(A_{2,11} = \{a_2, a_4, a_8\}\)

Now, our problem is to find \(S_{0,n+1}\) and \(A_{0,n+1}\)

If \(a_k \in A_{i,j}\), we can independently find \(A_{i,k}\), \(A_{k,j}\) repeatedly.

So, \(c_{i,j} = \max \left\{ c_{i,k} + 1 + c_{k,j} \mid i < j \right\} \)

We can use dynamic programming to compute \(c_{i,j}\). We create the matrix, and initialize the diagonal as 0. We only care about the upper part of the matrix and fill up the matrix diagonal by diagonal.

Greedy algorithm 1(earliest finish time)

\(a_m \in A_{i,j}\), if and only if \(f_m = \min \{ f_k : a_k \in S_{i,j} \} \)

1. \(a_m\) is an element of some max-size solutions(There could be some optional solutions).
2. \(S_{m}\) is empty.

Proof:
We do not need to consider $a_c$, because it does not affect $a_m$.

$a_A$ --- it cannot find better solution. Because any solution containing $a_A$ can contain $a_m$.

$a_B$ --- it cannot find better solution. $a_m$ wins slot 2. It seems that $a_B$ wins slot 1. However, $f_m = \min\{f_k : a_k \in S_{i,j}\}$, so actually slot 1 does not exist.

GreedySel

\[
\begin{align*}
A &= \{a_1\} \\
i &< 1 \\
\text{for } m &< 2 \text{ to } n \\
\quad \text{if } s_m &\geq f_i \\
\quad A &= a \cup \{a_m\} \\
\quad i &= m \\
\end{align*}
\]

Greedy algorithm 2 (shortest time span)

Pick the shortest element from the activities set in which each activity’s start time is after the finish time of each activity in the solution.
Conclusion:

- The solution found by greedy algorithm is generally suboptimal
- Sometimes the solution is optimal
- Whether or not a solution is optimal can be proved by dynamic programming or other algorithms
- The point is to design a good criterion to cut the branches