DYNAMIC PROGRAMMING, PART 3: SPACE-SAVING TECHNIQUES

Marcus Ritt

CMP 601 – Algorithms and Theory of Computation —

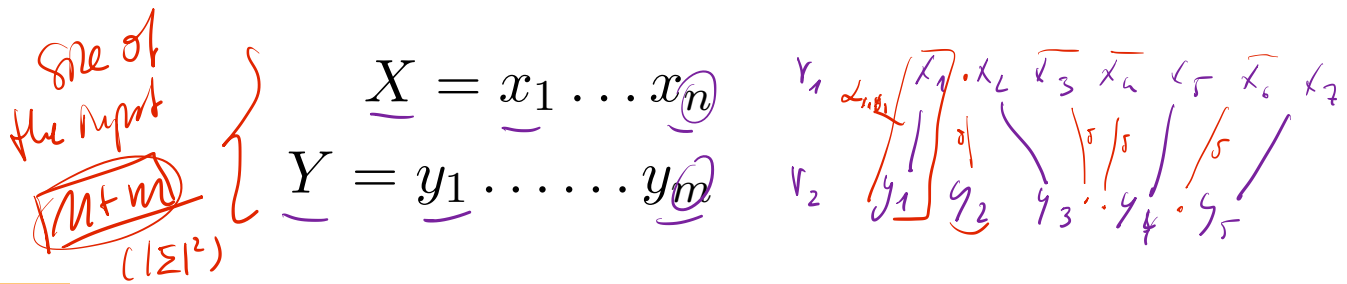
<2020-05-21 qui>

Outline

1. Minimum cost sequence alignment
2. Space-saving for MCSA
3. Lagniappe

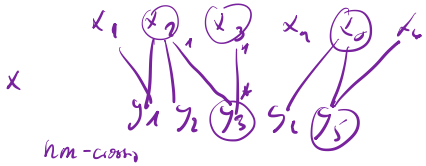
- Given two sequences over some alphabet Σ

$$X = x_1 \dots x_n$$
$$Y = y_1 \dots \dots y_m$$



- find *alignment*: non-crossing matchings of X's to Y's characters
- a minimal *cost*
  - matching $c, d \in \Sigma$ costs α_{cd}
  - unmatched characters are "gaps" and cost δ

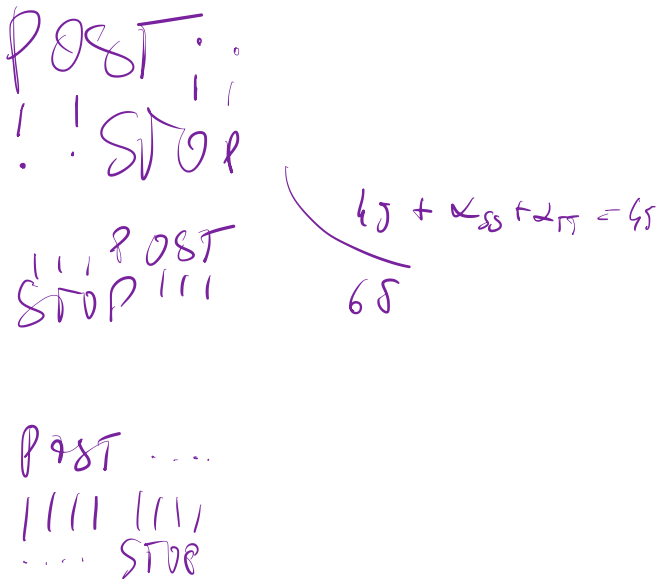
Ex: solve the problem with multiple matchins (but still non-crossong way).



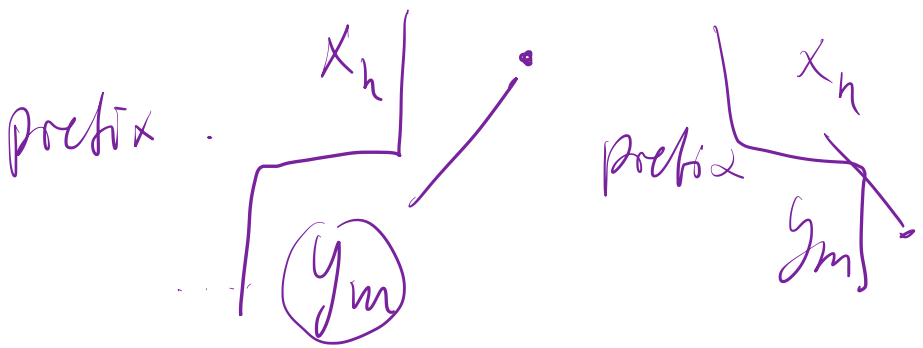
Example

- POST and STOP
- Empty matching costs?

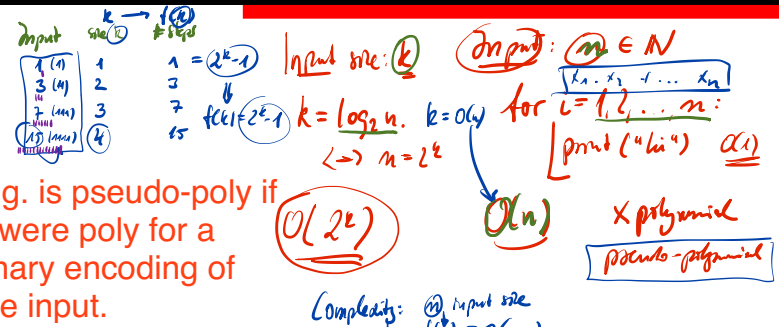
$UB = (n+m)\delta$



- Look at the last two characters x_n and y_m, either
  - they are matched: cost α_{x_n,y_m}, subproblems: prefixes $X_{1:n-1}, Y_{1:m-1}$
  - x_n is matched: then y_m must be free (why?): cost δ, prefixes $X_{1:n}, Y_{1:m-1}$
  - y_m is matched: then x_n must be free: cost δ, prefixes $X_{1:n-1}, Y_{1:m}$



Resultant recursion

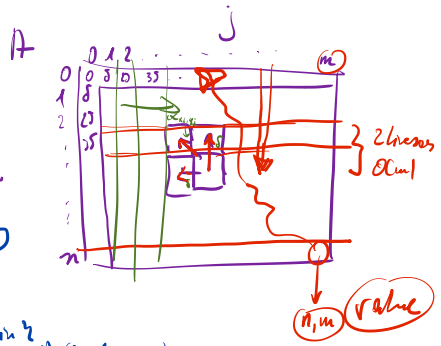


- That gives us a recursion.
- Let $A(i,j)$ be the cost of the best alignment of $X_{1:i}$ and $Y_{1:j}$.
- We want: $A(n,m)$.
- A satisfies

$$A(i,j) = \begin{cases} \min\{\alpha_{x_i,y_j} + A(i-1,j-1), \delta + A(i-1,j), \delta + A(i,j-1)\}, & i > 0 \wedge j > 0, \\ \delta(i+j), & ij = 0. \end{cases}$$

- This would be a backward DP

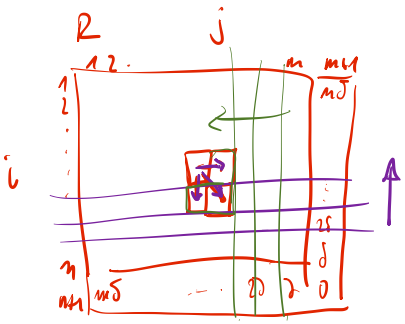
Space complexity: $O(nm)$
Time complexity: $O(nm)$



Forward DP

- We can also compute this solution in a forward manner
- Compare x_1 and y_1, same trichotomoy, subproblems are now suffixes
- Let $R(i, j)$ be the cost of the best alignment of $X_{i:n}$ and $Y_{j:m}$.
- R satisfies

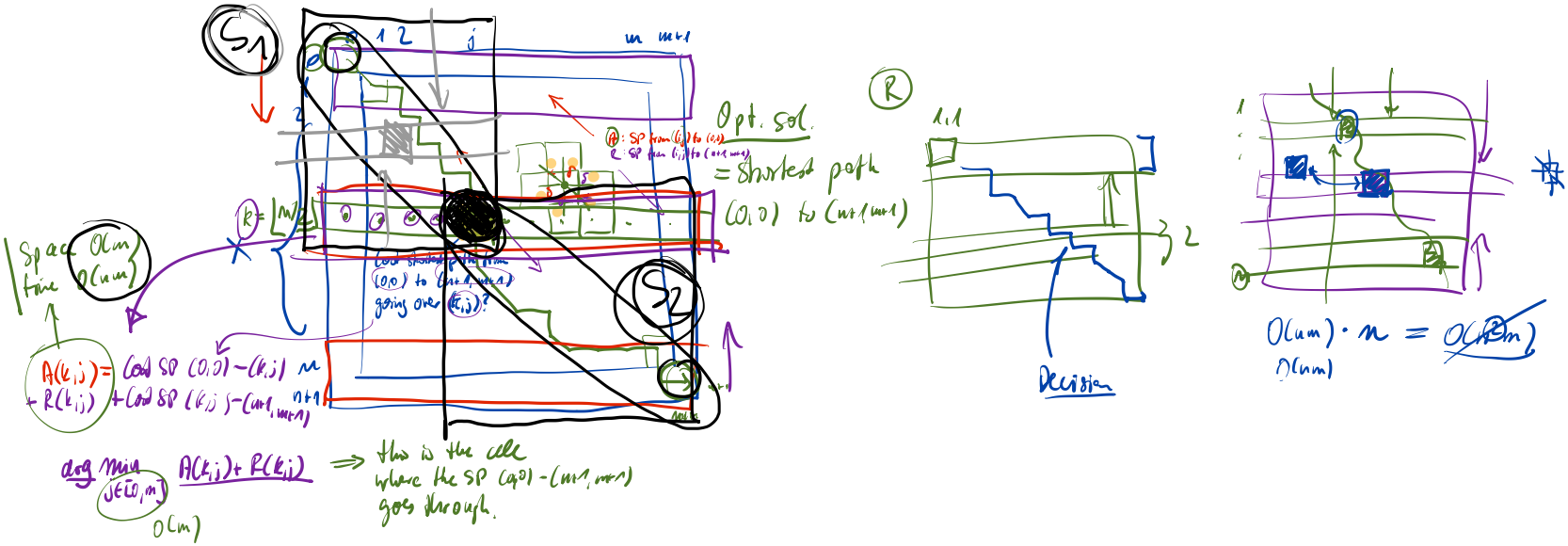
$$R(i,j) = \begin{cases} \min\{\alpha_{x_i,y_j} + R(i+1,j+1), \delta + R(i+1,j), \delta + R(i,j+1)\}, & i \le n \wedge j \le m, \\ \delta((n+1-i)+(m+1-j)), & (n+1-i)(m+1-j) = 0 \end{cases}$$



Space and time complexity

- Both take space $O(nm)$
- Both take time $O(nm)$, too
- Assume (wlog) $m < n$: space can be reduced to $O(m)$ by propagating only lines
- But we lose the solution

- Main problem now: how to recover a solution



Insight

- The solution is a shortest path from $(0,0)$ to $(n+1, m+1)$
- If we fix some line k must pass through some (k, j), for $j \in [0, m+1]$
- For shortest paths $C((i,j),(i',j'))$ are

$$A(i,j) = C((0,0),(i,j))$$
$$R(i,j) = C((i,j),(n+1,m+1))$$

- We check them all and recursively compute the shortest path

Space: $O(m)$

1. Compute $A(k,j)$ and $R(k,j)$ for $j \in [0, m+1]$ in O(nm) by propagating rows. space $O(m)$
2. Select j such that $A(k,j) + R(k,j)$ is minimal. Then cell ~~(i,k)~~ (k,j) is on the shortest path.
3. Recursively find the shortest path from $(0,0)$ to ~~(i,k)~~ (k,j) and the one from ~~(i,k)~~ (k,j) to $(n+1, m+1)$.
4. Join the shortest paths.

Costs

- Space cost now is $O(m)$
- Time

$$T(n, m) = T(\lfloor n/2 \rfloor, j) + T(\lceil n/2 \rceil, m + 2 - j) + c'nm$$
$$= O(nm).$$

Proof by substitution

$$
\begin{aligned}
T(n,m) &\leq T(n/2,j) + T(n/2, m+2-j) + c'nm \\
&\leq c/2nj + c/2n(m+2-j) + c'nm \\
&(c/2 + c')nm + cn && \text{for } m \geq c \\
&(c/2 + c' + 1)nm && \text{for } c \geq 2c' + 2 \\
&cnm.
\end{aligned}
$$

A memetic algorithm for the Cost-oriented Robotic Assembly Line Balancing Problem

Jordi Pereira[a*], Marcus Ritt[b] and Óscar C. Vásquez[c]

[a] *Universidad Adolfo Ibáñez, Faculty of Engineering and Sciences , Viña del Mar, Chile*
[b] *Instituto de Informática, Universidade Federal do Rio Grande do Sul, Porto Alegre, Brazil*
[c] *Industrial Engineering Department, University of Santiago of Chile, Chile*

(*May 31, 2017*)

Problem is . . .

- Tasks $t_1, \ldots, t_{|T|}$
- Set of robots R with fixed cost c_r^f, $r \in R$
- Task t_j takes time p_{r,t_j} on robot r.
- Variable costs are c_{r,t_j}^v for the tasks.
- A cycle time c for the production line.
- Buy robots and assign tasks to them.

Theorem 3.1. *The cRALBP with a given specific order to execute the tasks admits a polynomial time algorithm in the input size.*

Proof. A polynomial time algorithm for the cRALBP with a given specific order to execute the tasks is obtained by a more efficient dynamic program, deciding for each station which prefix of the remaining tasks and which robot will be assigned to it. Let $\mathrm{C}(i)$ be the minimal total cost of executing tasks $t_i, t_{i+1}, \ldots, t_{|T|}$. These costs satisfy the recurrence

$$\mathrm{C}(|T|+1) = 0,$$

$$\mathrm{C}(i) = \min_{\substack{r \in R, k \in \{i+1,\ldots,|T|+1\}: \\ \sum_{i \le j < k} p_{rt_j} \le c}} \left\{ c_r^f + \sum_{i \le j < k} c_{rt_j}^v + \mathrm{C}(k) \right\} \tag{21}$$

To see the correctness of recurrence (23), note that for initial task i some prefix $T_i, \ldots, T_{k-1}$, $k > i$, of the tasks must be executed on the first station by some robot r. This has fixed cost c_r^f and variable cost $\sum_{i \le j < k} c_{rt_j}^v$. The station is then closed, and we are left with the residual problem of executing the remaining tasks $k, k+1, \ldots$ starting at an free station. The optimal cost is obtained by minimizing over all feasible choices of r and k, which must satisfy $\sum_{i \le j < k} p_{rt_j} \le c$. Correctness thus follows from a backward induction over i.

The corresponding dynamic programming table has $O(|T|)$ entries and each entry takes time $O(|R|\,|T|)$ to compute. Thus the optimal assignment of robots and tasks to stations can be found in time $O(|T|^2\,|R|)$. □