ICPC Europe Contests

icpc international collegiate programming contest

icpc global sponsor programming tools

icpc diamond multi-regional sponsor

Central Europe Regional Contest

ICPC CERC 2022

Solution Presentation

University *of Ljubljana*
Faculty *of Computer and Information Science*

Ljubljana, 27. 11. 2022

L - The Game

Simulate the described card game.

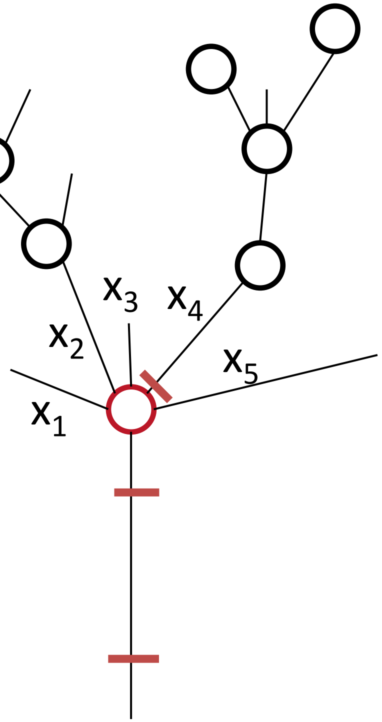
- maintain lists of cards:
  - rows, hand, deck
- careful implementation
  - prioritize backward moves
  - choose best regular move
    - sort by (abs. difference, hand, row)

```
rows:                  hand: 16, 55, 70, 67, 13, 9, 12, 40
1, 3
1, 7, 8, 9             deck: 14, 90, 31, 33, …
100, 60, 70
100
```

D - Deforestation

Cut a tree into parts of size at most W using fewest cuts.

- recursive input
- greedy strategy
- prune the tree from leaves towards the root
  - cut off part of size W
- node with "stumps" of sizes $x_i < W$
  - $\sum x_i > W$ --> cut off largest stumps
  - $\sum x_i \leq W$ --> cut up parent branch
- solve(a) … optimal cutting of subtree rooted in a
  - minimum number of cuts
  - remaining size of the stump
- O(n log n)
  - challenge: O(n)



E - Denormalization

Undo normalization of a list of small integers.

- too many possible vector lengths … $d = \surd(\sum a_i^2)$
- intermediate step: normalize to min=1 (divide by k=min(a))

a =	5	6	10	15	30	6
min =	1.000	1.196	1.993	2.993	5.978	1.196
x/norm =	0.138	0.165	0.275	0.413	0.825	0.165

- reverse direction
  - norm -> min: divide by min(x)
  - min -> a:
    - $a_i = \min_i \cdot k$, $1 \le k \le 10\,000$
    - find integer k that yields a_i that are closest to integer values and in range
    - O(AN)
- making an assumption about the value of min(a) or max(a)

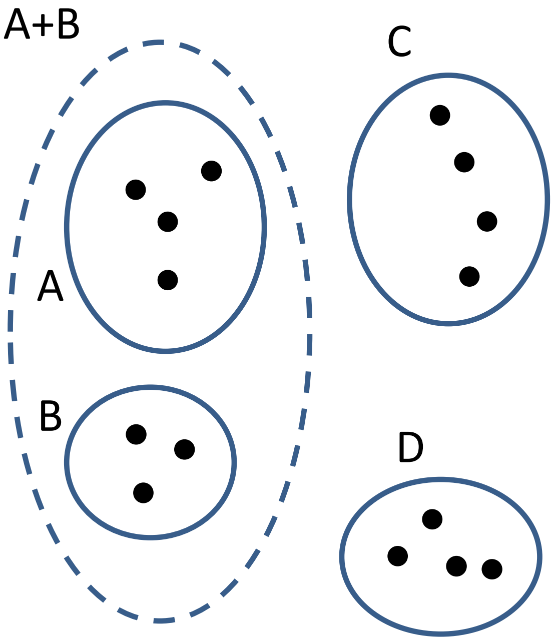
C - Constellations

Compute hierarchical clustering of points using squared Euclidean distance.

- brute-force: ~~$O(n^5)$~~ $O(n^3)$
- constellation … list of stars
- priority queue of potential constellations
  - (distance, min(a,b), max(a,b))
- merge, update distances

$$d'(A,B) = \sum_a \sum_b \|a-b\|^2$$
$$d'(A+B,C) = d'(A,C) + d'(B,c)$$

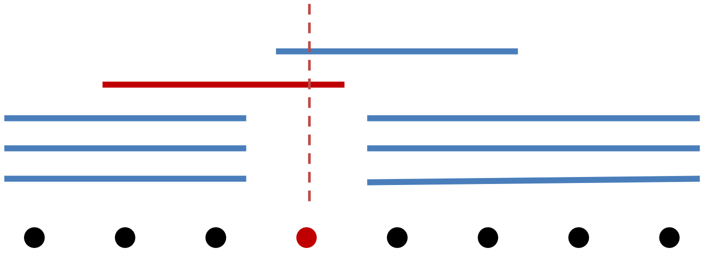
- $O(n^2 \log n)$
  - form O(n) constellations
  - update O(n) distances in O(log n)



G - Greedy Drawers

Construct a counterexample for a greedy assignment of notebooks to drawers.

- does a notebook fit into a drawer?
  - horizontal orientation



- possible counterexample:
  - notebooks of dimensions (1,x), (2,x-1), …, (x,x)
  - a drawer can contain a range of notebooks
  - 50% chance of suboptimal assignment
  - repeat the pattern
- prob. of success (greedy finds suboptimal solution):
  - single case: $p_1 = 1 - 0.5^{(150/8)}$
  - all 20 cases: $p = p_1^{20} = 99.995\%$

K - Skills in Pills

Find an arrangement with a minimum number of pills that avoids taking two pills on the same day.

- if we could take both pills on the same day
  - take a pill as late as possible (pill A every k-th day and B every j-th)
- resolve first “collision”
  - shift one of the pills one day back; which one?

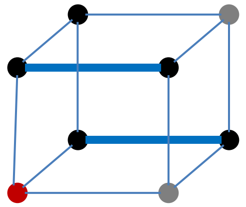
e.g. A=2, B=3, N=8

	A	B	A		AB			
take A early				A	B	A		6 pills
take B early				B	A		AB	7 pills

- dynamic programming
  - f(n, AB) … min number of pills taken in the remaining n days if we take pills A and B in this order in preceding two days
  - compute next collision
  - O(n)
- challenge: sublinear greedy solution

B - Combination Locks

Find the winner in a two-player game with non-repeating states



- Hypercube graph
  - node = difference pattern, forbidden nodes
  - can move to any adjacent node
  - bipartite
- alternatingly building a simple path in a graph
- possible strategy: following edges in a maximum matching
- maximum matching that doesn’t include the starting node?
  - Yes: Bob can follow matched edges
    - stuck at unmatched node -> there would exist an augmenting path
  - No: Alice can follow matched edges
    - stuck at unmatched node -> flip edges, get an unmatched start node

F - Differences

Find a string with Hamming distance K to all other strings.

- $O(n^2)$ too slow
- precompute sets of strings that have character c at position j … f(j,c)
- sets of strings differing from string S_x at each position j (union)
- Hamming distances from S_x
- speed-up:
  - use bit masks to represent sets of strings?
  - use polynomial hashes ... O(nm)
    - e.g., $f(0,A) = (p^0+p^2)$ % mod, $g(j) = \sum f(j,A)$
    - S_x … $\sum_j g(j) - f(j, S_{x,j})$ should be equal to $\sum_i Kp^i - p^x$

S_x=CA, S={AB, BA, AB, CA, CA, CC}

j=0	j=1
A: {0,2}	A: {1,3,4}
B: {1}	**B: {0,2}**
C: {3,4,5}	**C: {5}**
{0,1,2}	{0,2,5}

d = [2, 1, 2, 0, 0, 1]

goal: [K, K, K, 0, K, K]

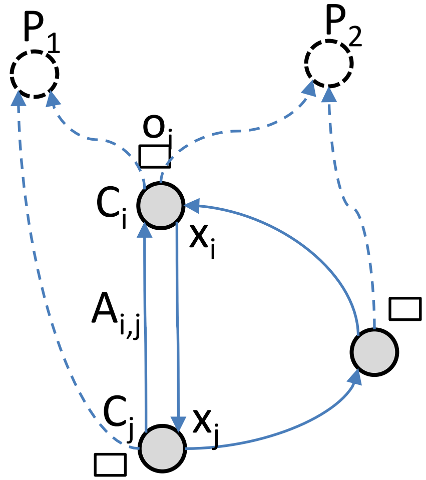
I - Money Laundering

Compute individual’s ownership shares in a network of company ownerships.

- simulate redistribution
  - $x = [x_1, \ldots, x_n]^T$ … vector of company incomes
  - redistribution matrix A, $x' = Ax$
  - $A_{i,j}$ … share received by i from j
  - A^k converges to 0
- accumulate output values
  - $o = x + Ax + A^2x + \ldots$

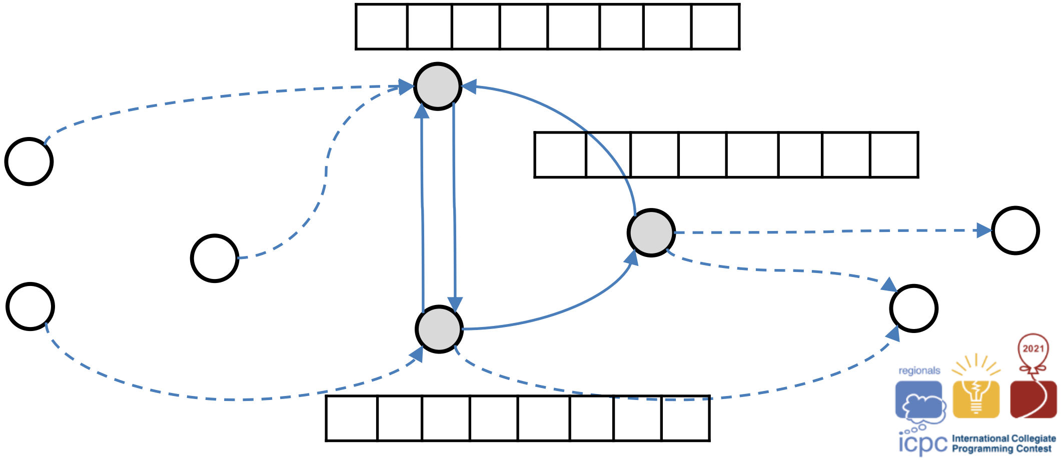
  a) geometric series
  - $o = (I - A)^{-1} x$
  - inverse (Gauss–Jordan elimination)

  b) power method
  - $y = [x_1, \ldots, x_n, o_1, \ldots, o_n]^T$, $B=\begin{bmatrix} A & 0 \\ I & I \end{bmatrix}$
  - $y' = By$, B^{big} … exponentiation by squaring



I - Money Laundering

- industrial sectors = strongly connected components
  - Tarjan, Kosaraju, …
  - small!
  - ownership structure (income) from preceding companies
    - matrix X: $X_{i,j}$ … income received by company i from company j
  - extract submatrix of X relevant to the SCC (dim. S x C)
  - propagate income within SCC
  - distribute to persons and companies
- O(C/S S^3 + K C)
  - C … companies
  - K … edges
  - S … max size of SCC



J - Mortgage

Given the monthly incomes, compute the largest monthly payment that you can afford in the range of months [L … R].

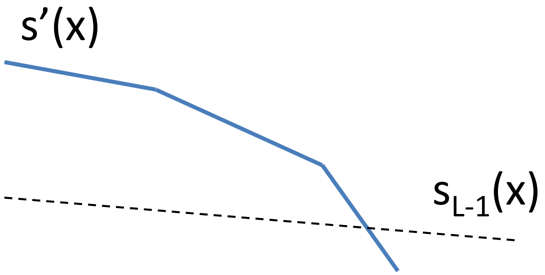
a) *algebraic approach*

- consider a fixed payment x
  - b_j = balance on day j
  - range minimum query (tree)
- unknown x?
  - $s_j(x)$ is a linear function of x
  - store lower envelopes s'(x) of $s_j(x)$ in each node
  - binary search for x in each range: $s(x) \geq s_{L-1}(x)$
    - s_{L-1} is the flattest
  - $O(n \log n + m \log^2 n)$

$$s_j = \sum_{i=1}^{L} -jx$$

$$b_j = s_j - s_{L-1} = \sum_{i=L}^{j} a_i - (j - L + 1)x$$

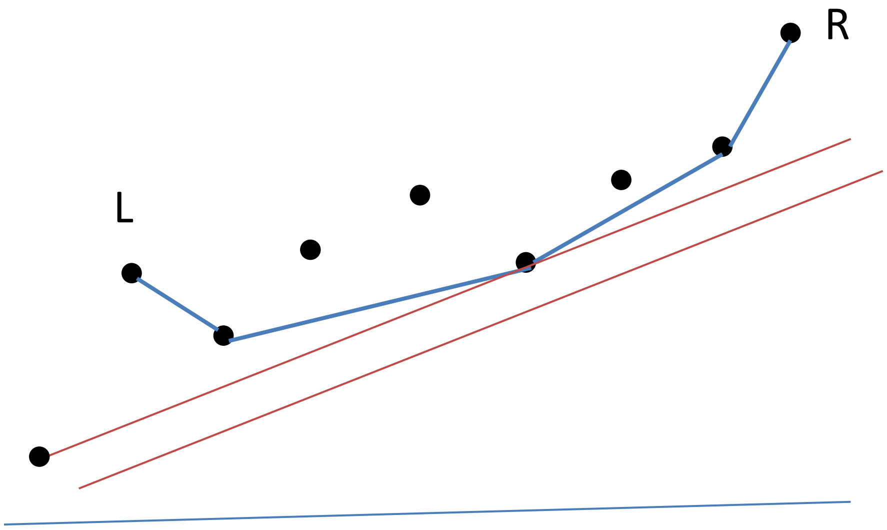
$$b_j \geq 0 \Rightarrow s_j \geq s_{L-1} \Rightarrow \min_{j \in [L,R]} s_j \geq s_{L-1}$$



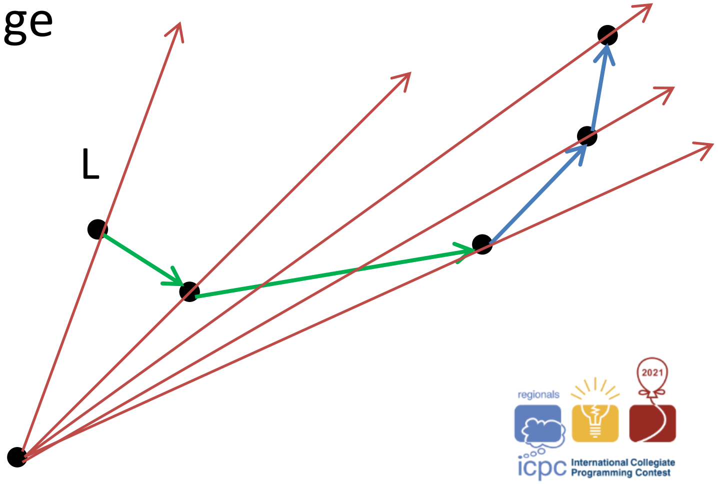
J - Mortgage

b) *geometric approach*

- points (i, c_i), $c_i = \sum_{j=1..i} a_i$
- query [L, R] … steepest line originating from L-1



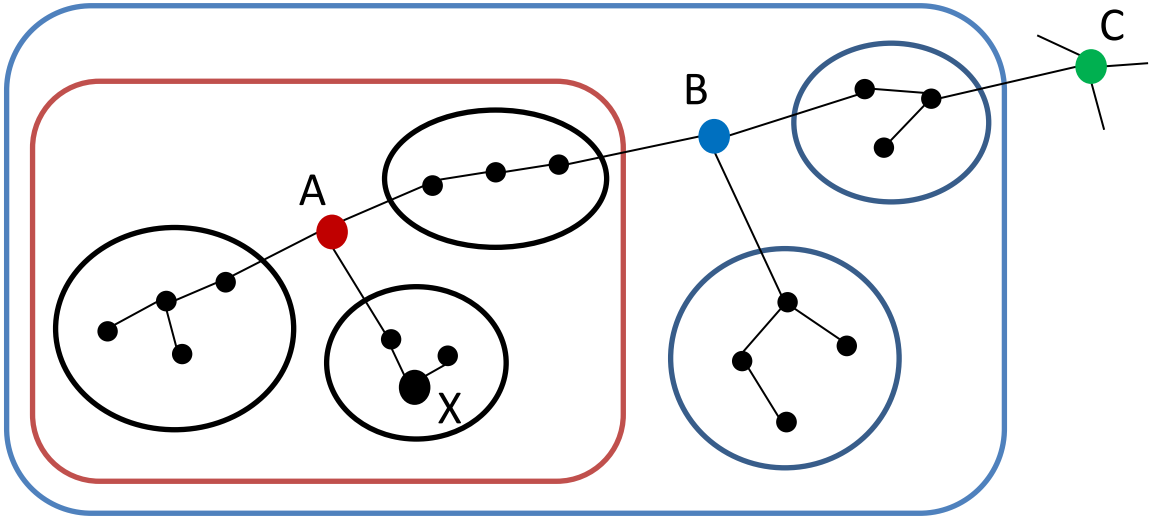
- partition points into groups
  - lower hull
  - tree structure of groups
    - O(n) groups overall
    - O(log n) groups cover every query range
- binary search in a group
  - max prefix of the hull with segments that are clockwise to the line from L-1
- careful with overflows
- $O(n \log n + m \log^2 n)$

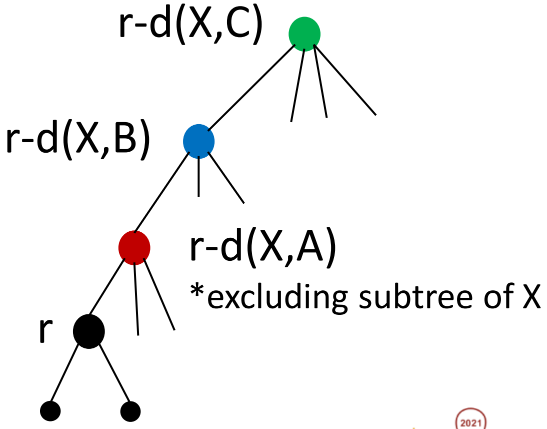


A - Bandits

Protect nodes in a tree at a distance at most r from X and answer queries about the level of protection of road Y.

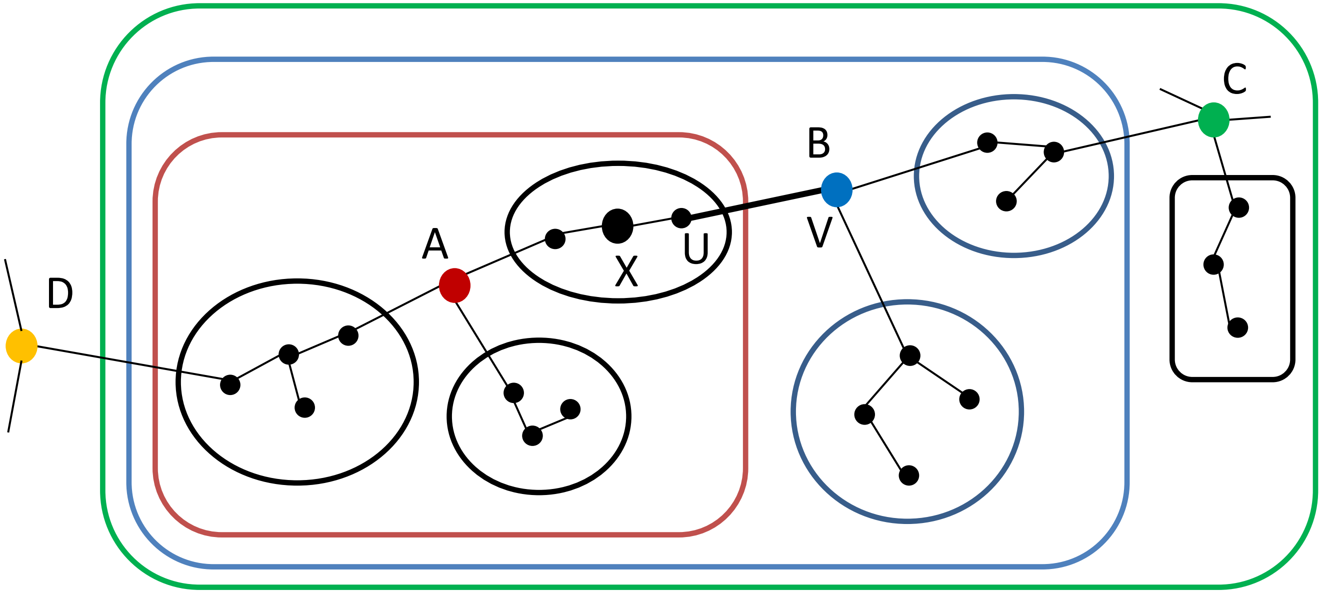
- centroid decomposition
- new security contract at X with radius r
  - mark parts of the tree as protected … O($\log^2$ n)
  - store affected distance in a tree structure

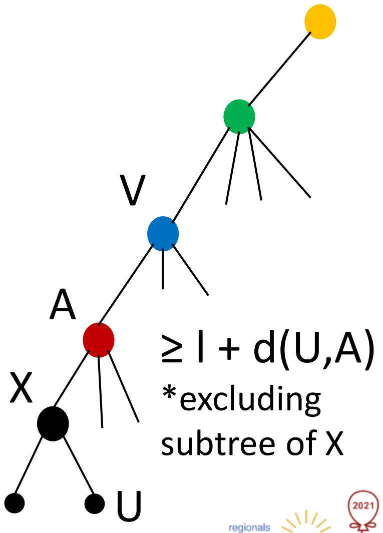




A - Bandits

- coverage of edge U-V with length l
  - V … more important centroid
  - protection originating from subcomponents of V (U, X, A), entering via U
    - # of markings ≥ l + d(U,A) [excluding subtree of X]
  - protection from large components (e.g. C) containing U and V
    - # of markings ≥ l + min(d(U,C), d(V,C)) [excluding subtree of B]
- O(Q $\log^2$ N)

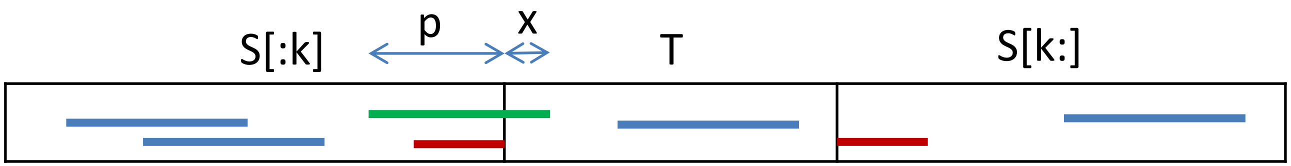




H - Insertions

Insert string T into S to maximize the number of patterns P.

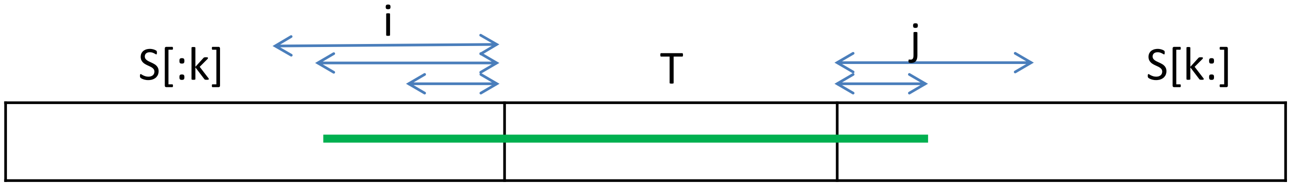
- consider all insertions after k chars
- count P in S and T, subtract those broken by insertion
  - KMP … locations of P in S and T



a) small patterns |P|≤|T|

- p = len. of longest prefix of P as a suffix of S[:k] (KMP search phase)
  - is there an appropriate suffix of P (of length x=|P|-p) in T?
    - len. of longest suffix of P ending in T[L] (z-algorithm) equal to L?
  - precompute matches for shorter prefixes (KMP fail. fun.)
- O(|S| + |T| + |P|)

H - Insertions

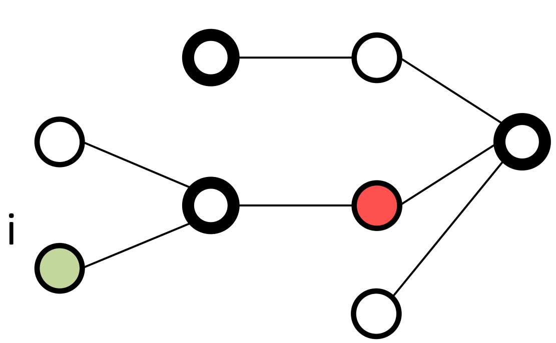


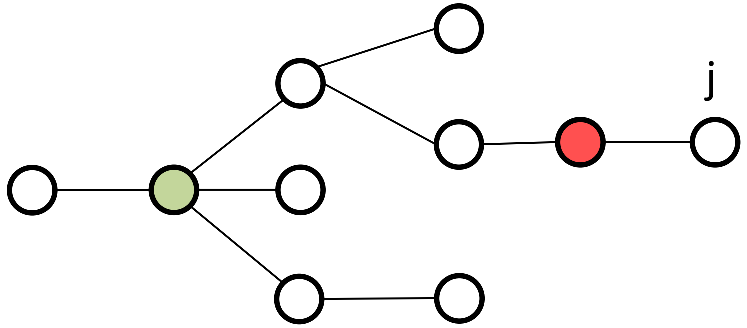
b) large patterns |P| > |T|

- can expand across entire T
  - does T match with shifted P? KMP search for T in P
- how many prefixes of P at the end of S[:k] match with suffixes of P at the start of S[k:]?
  - consider all pairs of shorter prefixes and suffixes … $O(|S|\cdot|P|^2)$
  - consider only shorter prefixes … $O(|S|\cdot|P|)$
    - as in the case for small patterns (z-algorithm)

H - Insertions

- trees of KMP failure functions f(i) of P and g(j) P^R





  - x(i,j) = number of matching nodes (correct sum of length) on paths from i and j to the root
  - $x(i,j) = x(i,g(j)) + match_j(i) = x(f(i),j) + match_i(j)$
  - precomputation … $O(|P|^{1.5})$
    - x(i, 0)
    - x(i’, j) for well-positioned special nodes i’ (including root)
      - subtrees of size sqrt(n)
    - x(i,j) … move towards root to first special node (≤ sqrt(n))
- $O(|S|+|T|+|P|^{1.5})$

The End