Query Optimization
Exercise Session 7

Andrey Gubichev

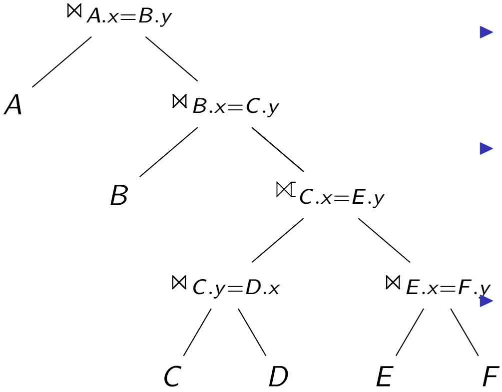
December 1, 2014

Enumerating Complementary Subgraphs

EnumerateCmp(G,S_1)
$X = \mathcal{B}_{\min(S_1)} \cup S_1$;
$N = \mathcal{N}(S_1) \setminus X$;
for all ($v_i \in N$ by descending i) {
    emit $\{v_i\}$;
    EnumerateCsgRec(G, $\{v_i\}$, $X \cup (\mathcal{B}_i \cap N)$);
}

- EnumerateCsg+EnumerateCmp produce all ccp
- resulting algorithm DPccp considers exactly #ccp pairs
- which is the lower bound for all DP enumeration algorithms

Homework

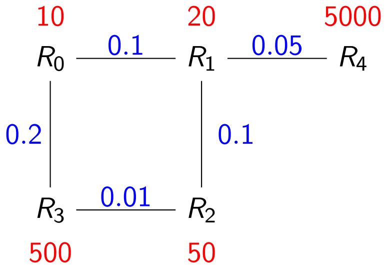


- Syntactic eligibility set - relations that have to be in the input
- Total eligibility set - captures also reordering restrictions, construct bottom-up
- Conflicts: $⟖_{C.x=E.y}$ and $\bowtie_{C.y=D.x}$, $⟖_{C.x=E.y}$ and $\bowtie_{B.x=C.y}$

Homework: Graph Simplification

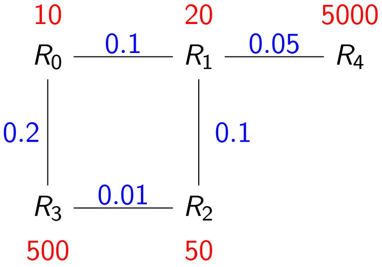
Important: consider all possible edge combinations, that is, $benefit(R_0 \bowtie R_1, R_0 \bowtie R_2)$ together with $benefit(R_0 \bowtie R_2, R_0 \bowtie R_1)$

Homework: Graph Simplification



- $benefit(R_0 \bowtie R_1, R_0 \bowtie R_3) = \frac{202}{300}$
- $b(R_0 \bowtie R_3, R_0 \bowtie R_1) = 300/202$
- $b(R_1 \bowtie R_2, R_1 \bowtie R_0) = 20/12$
- $b(R_3 \bowtie R_0, R_3 \bowtie R_2) = 2$
- $b(R_2 \bowtie R_3, R_2 \bowtie R_1) = 5/4$
- $b(R_1 \bowtie R_4, R_1 \bowtie R_0) = 500/251$
- $b(R_1 \bowtie R_4, R_1 \bowtie R_3) = 300/251$
- $R_3 \bowtie R_2$ before $R_3 \bowtie R_0$. Remove $R_3 - R_0$

Homework: Graph Simplification



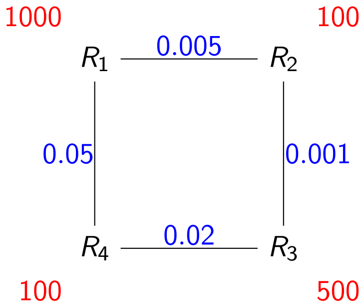
- $benefit(R_0 \bowtie R_1, R_0 \bowtie R_3) = \frac{202}{300}$
- $b(R_0 \bowtie R_3, R_0 \bowtie R_1) = 300/202$
- $b(R_1 \bowtie R_2, R_1 \bowtie R_0) = 20/12$
- $b(R_2 \bowtie R_3, R_2 \bowtie R_1) = 5/4$
- $b(R_1 \bowtie R_4, R_1 \bowtie R_0) = 500/251$
- $b(R_1 \bowtie R_4, R_1 \bowtie R_3) = 300/251$
- $b(R_0 \bowtie (R_3 \bowtie R_2), R_0 \bowtie R_1) = \frac{C((R_0 \bowtie (R_3 \bowtie R_2)) \bowtie R_1)}{C((R_0 \bowtie R_1) \bowtie (R_3 \bowtie R_2))} = 850/370$
- $b((R_2 \bowtie R_3) \bowtie R_0, R_2 \bowtie R_1) = \frac{C(((R_2 \bowtie R_3) \bowtie R_0) \bowtie R_1)}{C((R_2 \bowtie R_3) \bowtie R_1) \bowtie R_0)} = 1$
- $R_0 \bowtie R_1$ before $R_0 \bowtie (R_3 \bowtie R_2)$

Generating Permutations

ConstructPermutationsRec(P, R, B)
Input: a prefix P, remaining relations R, best plan B
Output: side effects on B

if $|R| = 0$ $\{$
  if $B = \epsilon \vee C(B) > C(P)$ $\{$
    $B = P$
  $\}$
$\}$ **else** $\{$
  for each $R_i \in R$ $\{$
    if $C(P \circ < R_i >) \leq C(P[1 : |P| - 1] \circ < R_i, P[|P|] >)$ $\{$
      ConstructPermutationsRec($P \circ < R_i >, R \setminus \{R_i\}, B$)
    $\}$
  $\}$
$\}$

Generating Permutations



- Keep current prefix and the rest of relations
- Extend the prefix only if exchanging the last two relations does not result in a cheaper sequence

Memoization

- DP: bottom-up construction of the join tree
- Memoization: top-down construction
- Memoize already generated join tree to avoid duplicate work
- Sometimes more efficient

Algorithms: Roadmap

- Deterministic
  - Exact (IKKBZ, DP, Permutations, Memoization,...)
  - Heuristics (GOO, MVP, Query Simplification,...)
- Probabilistic
- Hybrid

Random left-deep trees with cross products

- there are $n!$ trees (every tree - permutation)
- let's generate a random number in $[0, n![$
- *unranking* - for a generated number construct a tree
- *ranking* - for a tree define it's number

Generating random permutations

for each $k \in [0, n[$ **descending**
  $\text{swap}(\pi[k], \pi[\text{random}(k)])$

Array π initialized with elements $[0, n[$.
`random`(k) generates a random number in $[0, k]$.

Unranking

```
Unrank(n, r)
Input:  the number n of elements to be permuted
        and the rank r of the permutation to be constructed
Output: a permutation π
for each 0 ≤ i < n
  π[i] = i
for each n ≥ i > 0 descending {
  swap(π[i − 1], π[r mod i])
  r = ⌊r/i⌋
}
return π;
```

Random join trees with cross products

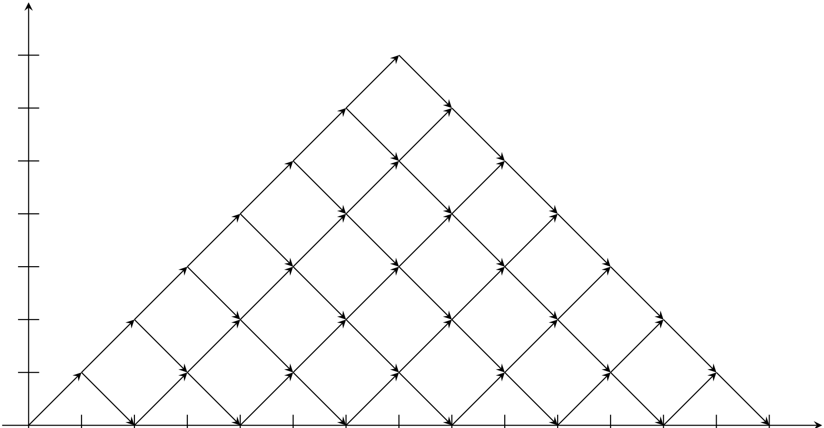
- Generate a tree, then generate a permutation: $C(n-1)$ trees, $n!$ permutations
- Pick a random number $b \in [0, C(n-1)[$, *unrank* b
- Pick a random number $p \in [0, n![$, *unrank* p
- Attach the permutation to the leaves

Unranking

- every tree is a word in $\{(,)\}$
- map such words to the grid, every step up is (, down)

Unranking

- every tree is a word in $\{(,)\}$
- map such words to the grid, every step up is (, down)



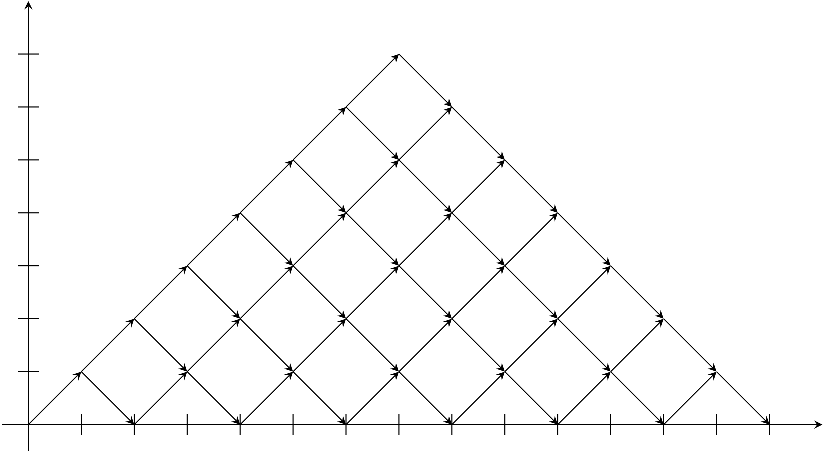
Unranking

- every tree is a word in $\{(,)\}$
- map such words to the grid, every step up is (, down)
- the number of different paths q can be computed (see lectures)
- Procedure: start in (0,0), walk up as long as rank is smaller than q. When it is bigger, step down, $rank=rank$-q

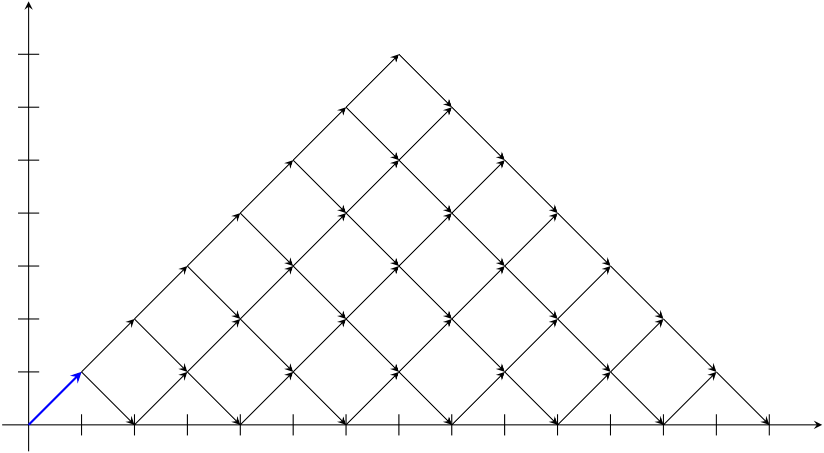
Example

- Bushy tree number 56, 8 leaves

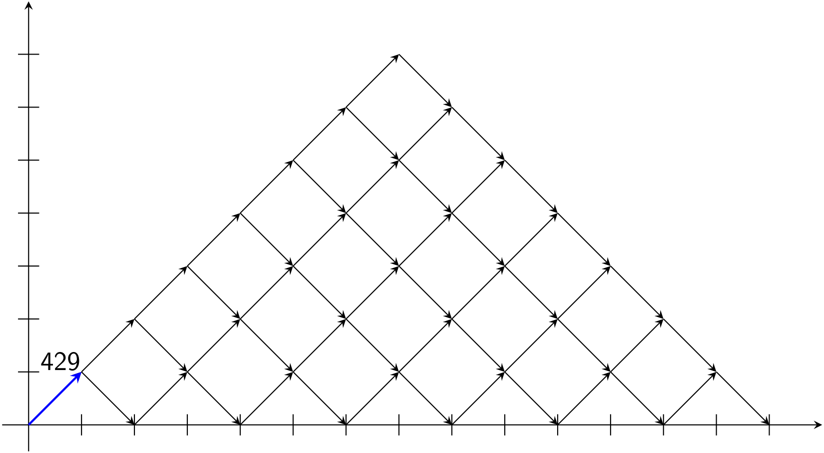
Random Join Tree Selection



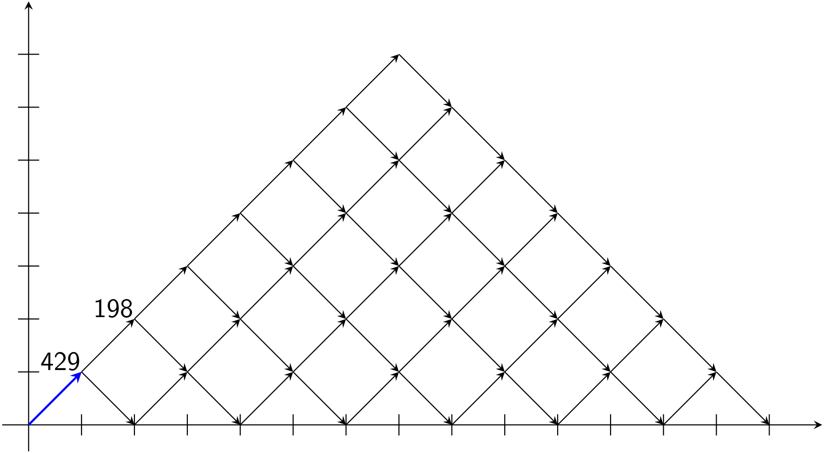
Random Join Tree Selection



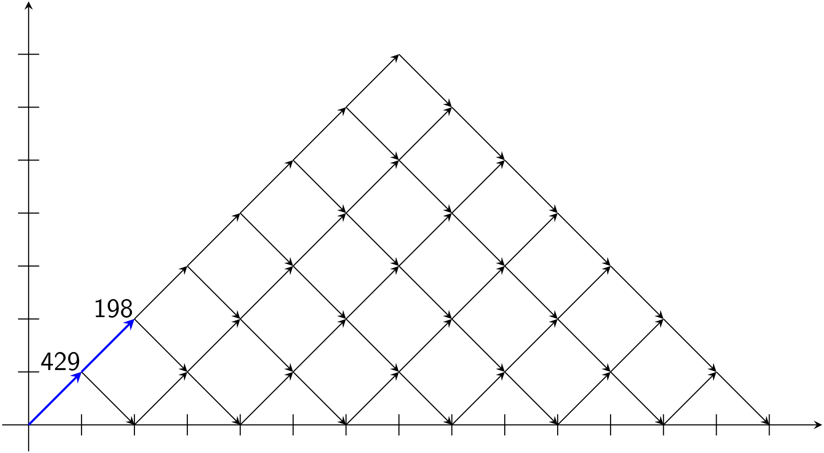
Random Join Tree Selection



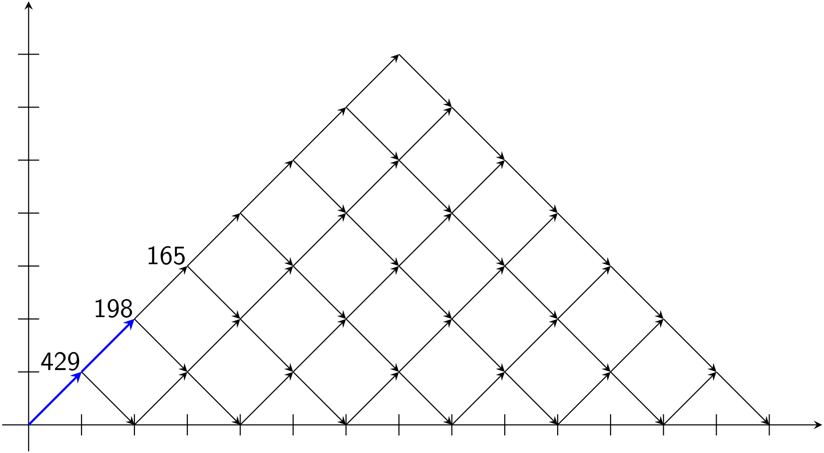
Random Join Tree Selection



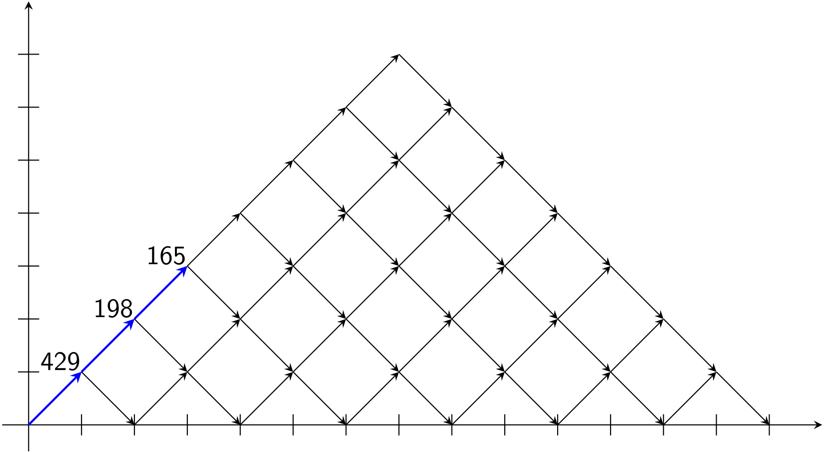
Random Join Tree Selection



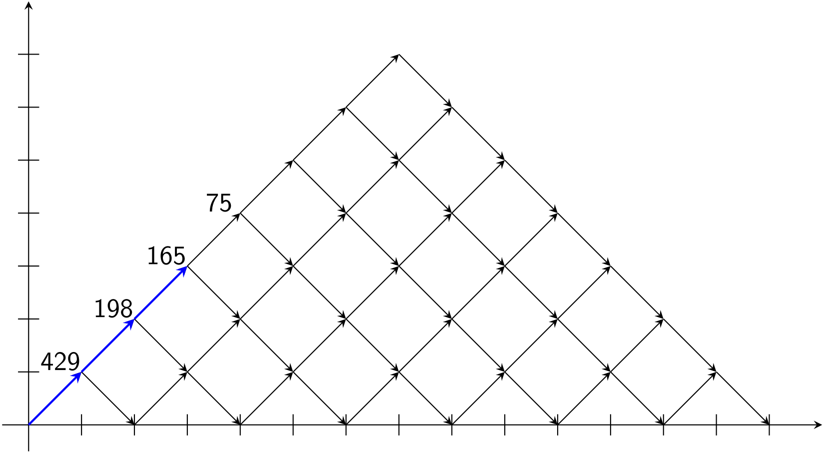
Random Join Tree Selection



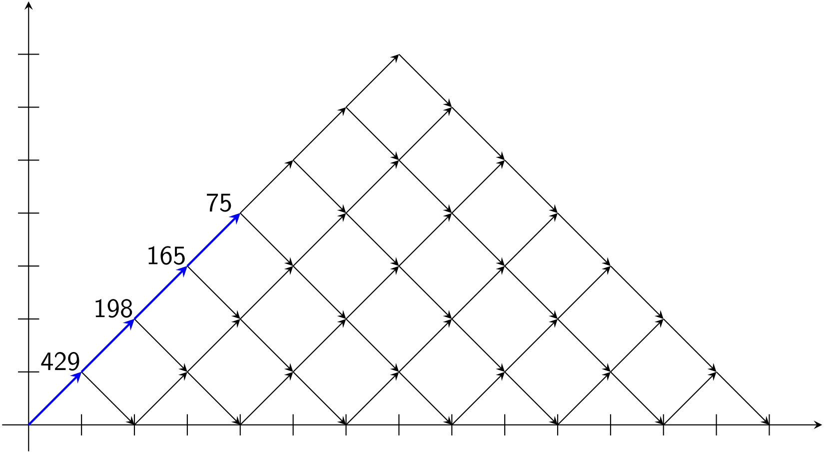
Random Join Tree Selection



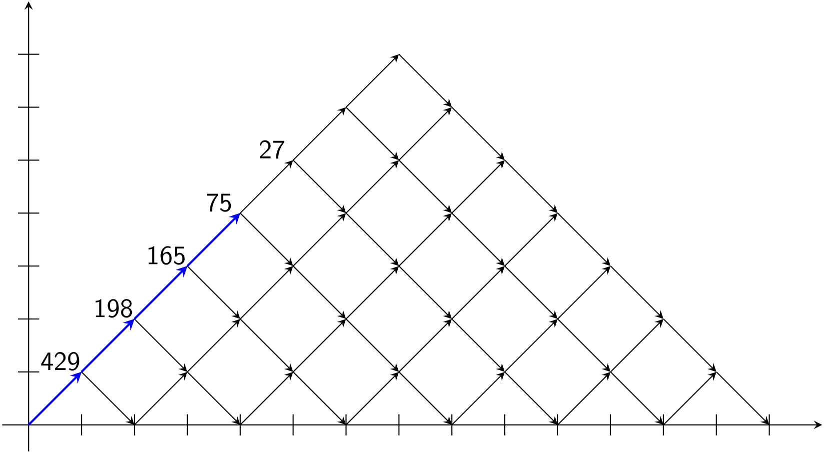
Random Join Tree Selection



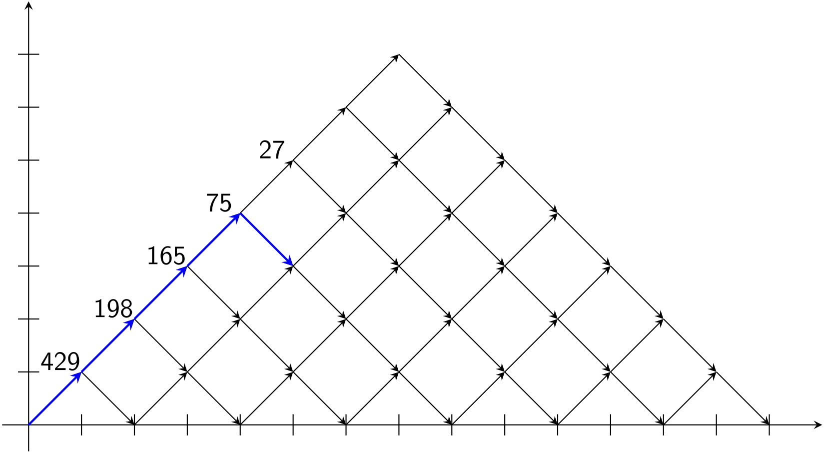
Random Join Tree Selection



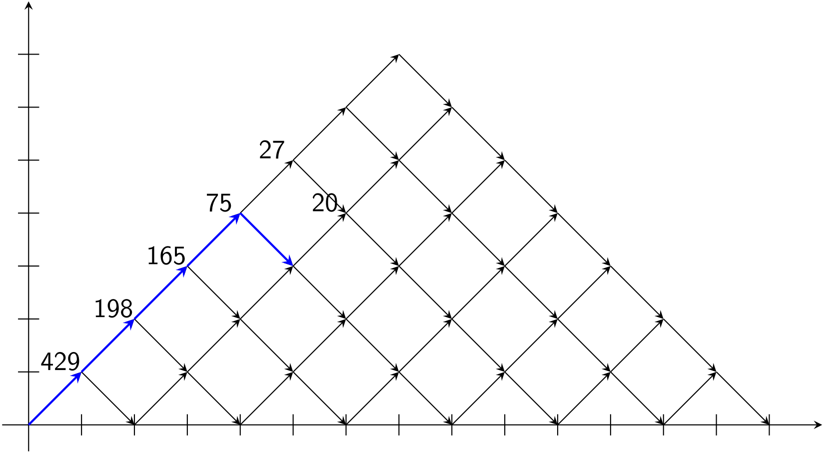
Random Join Tree Selection



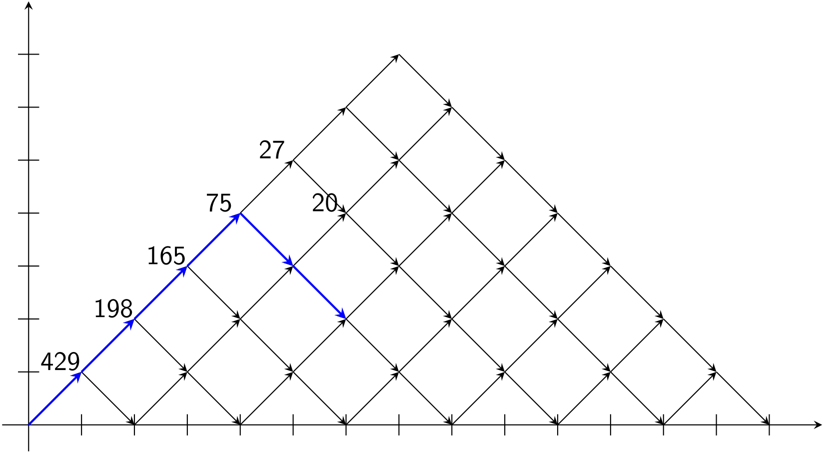
Random Join Tree Selection



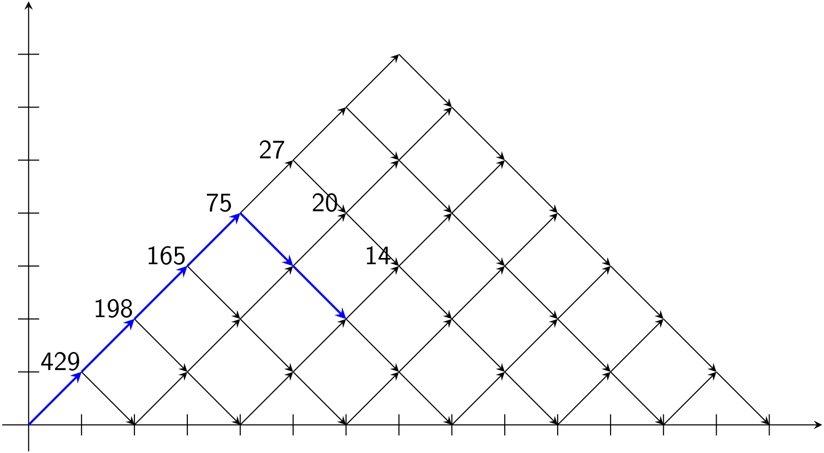
Random Join Tree Selection



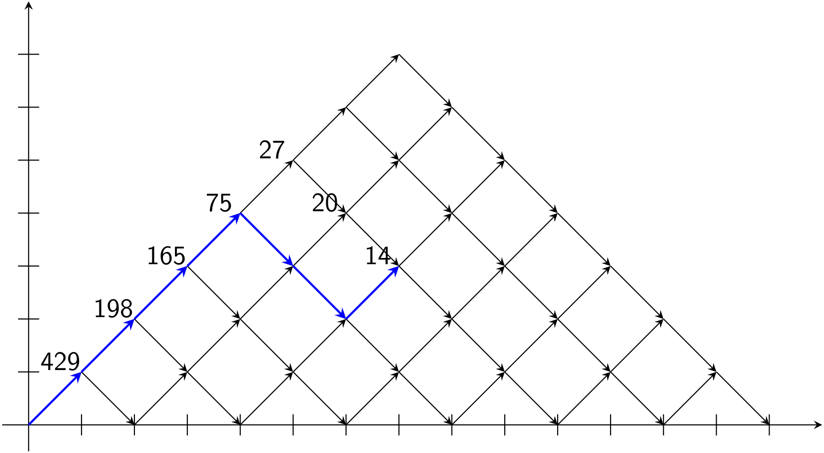
Random Join Tree Selection



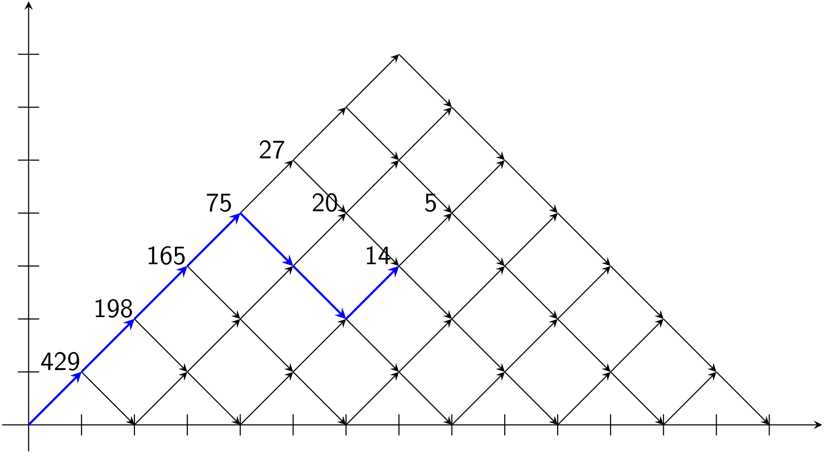
Random Join Tree Selection



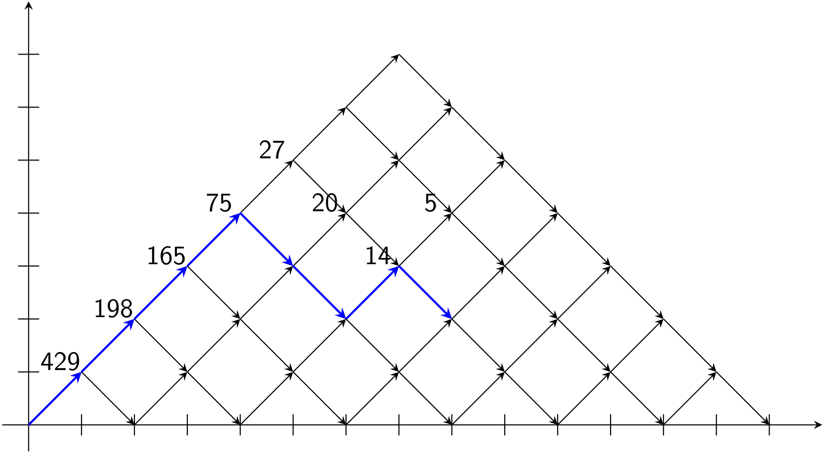
Random Join Tree Selection



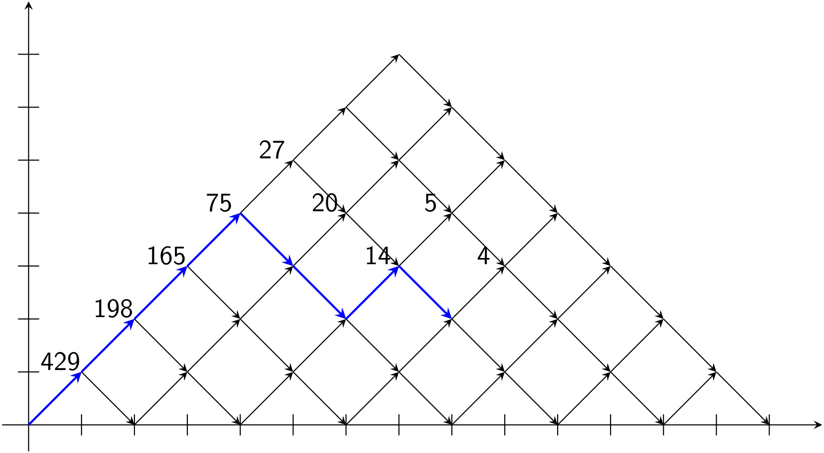
Random Join Tree Selection



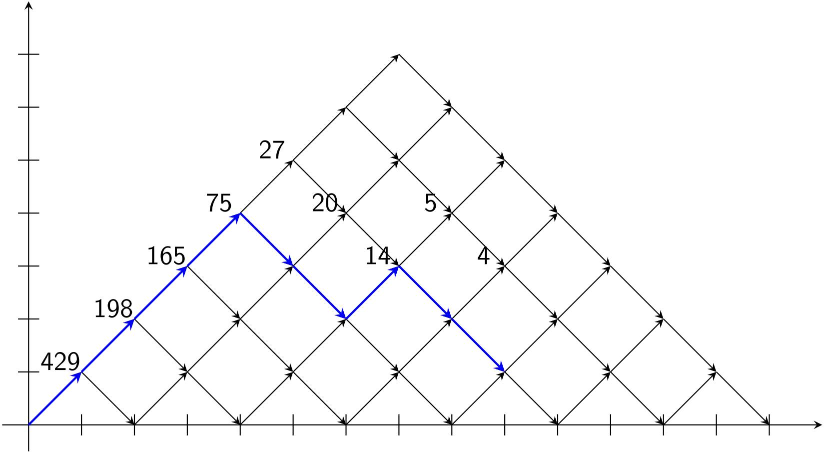
Random Join Tree Selection



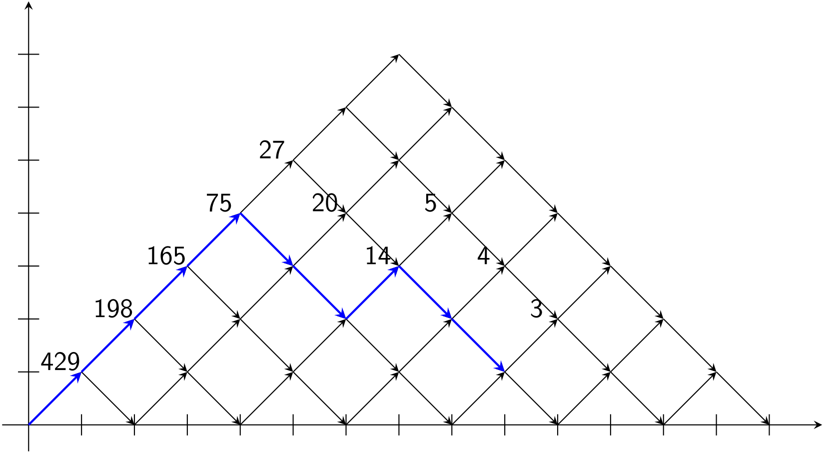
Random Join Tree Selection



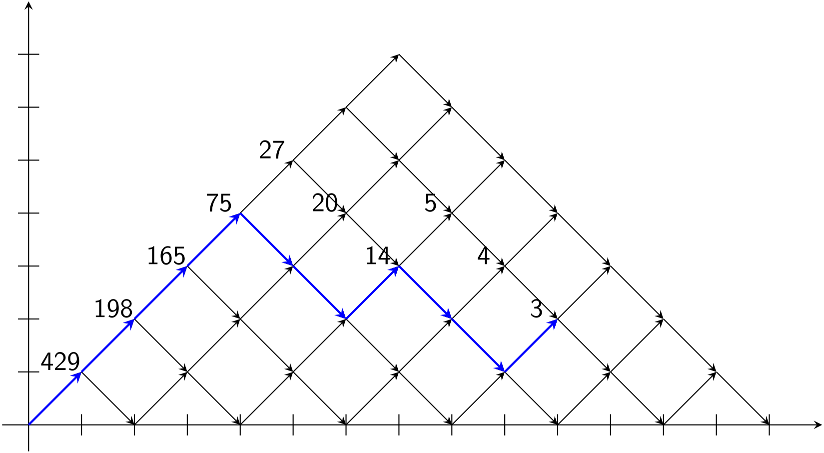
Random Join Tree Selection



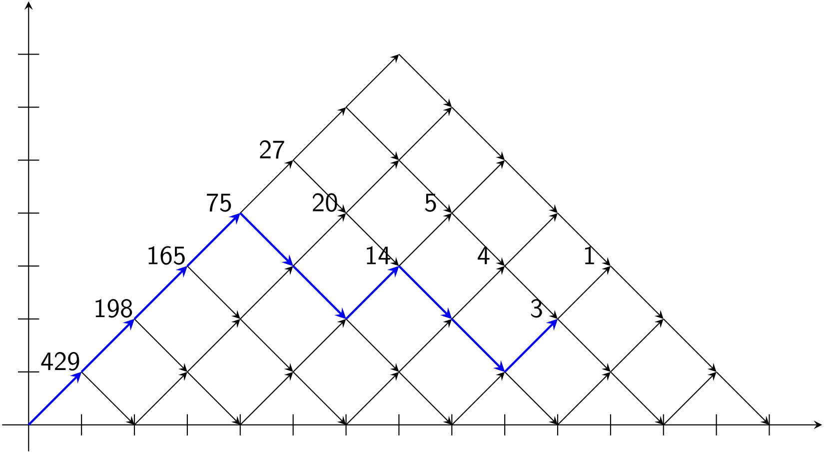
Random Join Tree Selection



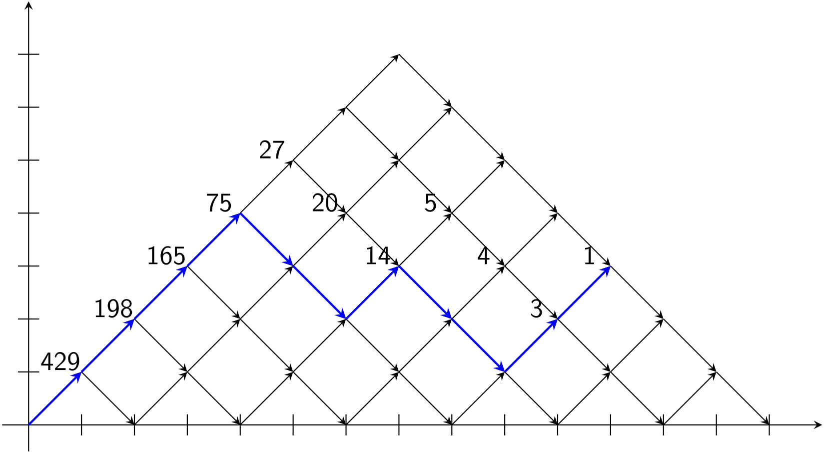
Random Join Tree Selection



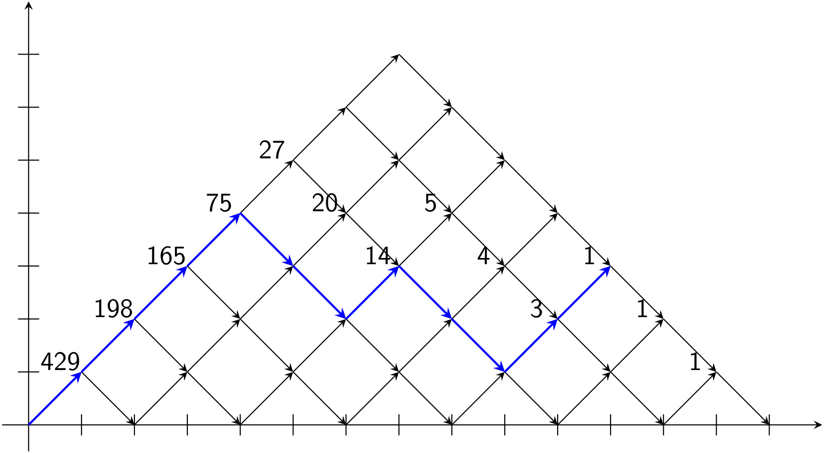
Random Join Tree Selection



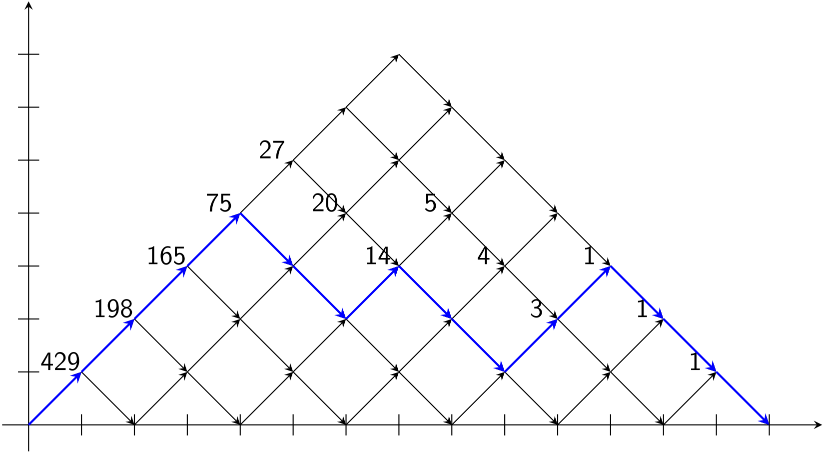
Random Join Tree Selection



Random Join Tree Selection



Random Join Tree Selection



Info

- Exercises due: 9 AM, December 8, 2014