On Compact Representations of All-Pairs-Shortest-Path Distance Matrices

Igor Nitto and Rossano Venturini

Dipartimento di Informatica

University of Pisa, Italy

Outline

- Problem statement
- A simple information-theoretic lower bound
- A first compact representation
- A more space-efficient representation

Problem Statement

- Given: an unweighted undirected graph
- Build: a static data structure encoding distance matrix **D**
  - **D**[u,v] := shortest path distance from u to v
- Using: O(minimum possible number of bits)...
- Supporting: access to any **D**[u,v] in O(1) time
- Model: Word RAM

Problem Statement

- Given: an unweighted
- Build: a static data st
  - D[u,v] := shortest path di ce from u to v
- Using: O(minimum possible number of bits)...
- Supporting: access to any D[u,v] in O(1) time
- Model: Word RAM

information-theoretic minimum

$$\frac{n(n-1)}{2} \sim O(1)\, bits\ per\ element$$

Comparison with related works

- Approximate Distance Oracles
  - Distributed data structures
  - Fast estimate of distances in small space
    - $O(n^{1+1/t} \log n)$ bits and $O(t)$ multiplicative error *[Thorup-Zwick, 2001]*

- Our result: exact distances in $O(1)$ time and $O(n^2)$ bits
  - First result offering such space/time tradeoff

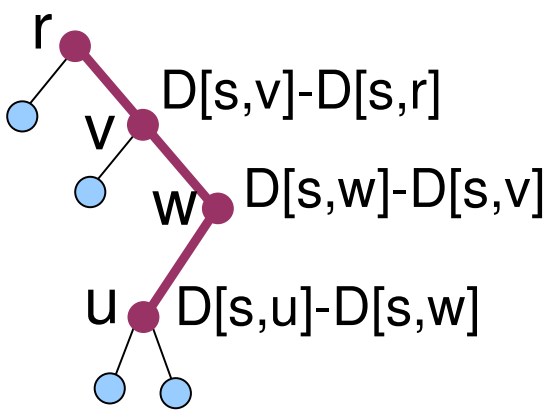
Single Source Distances Encoding

Let T be a spanning tree, rooted at an arbitrary node r

- (assume the graph connected)

Store separately each row **D**[s,...] in O(n) bits:

- Define the following labelling $\mathbf{T}_s$ of T:
  - $\mathbf{T}_s(r) = \mathbf{D}[s,r]$
  - $\mathbf{T}_s(u) = \mathbf{D}[s,u] - \mathbf{D}[s,f(u)]$ ($\in \{-1,0,1\}$) for $u \neq r$
- D[s,u] equals to *path-sum* at u in $\mathbf{T}_s$



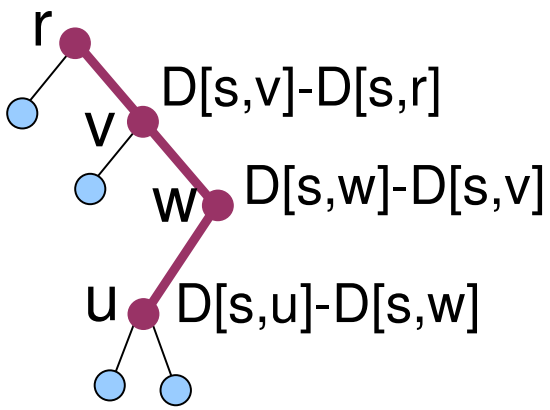
Single Source Distances Encoding

Let T be a spanning tree, rooted at an arbitrary node r

- (assume the graph connected)

Store separately each row **D**[s,...] in O(n) bits:

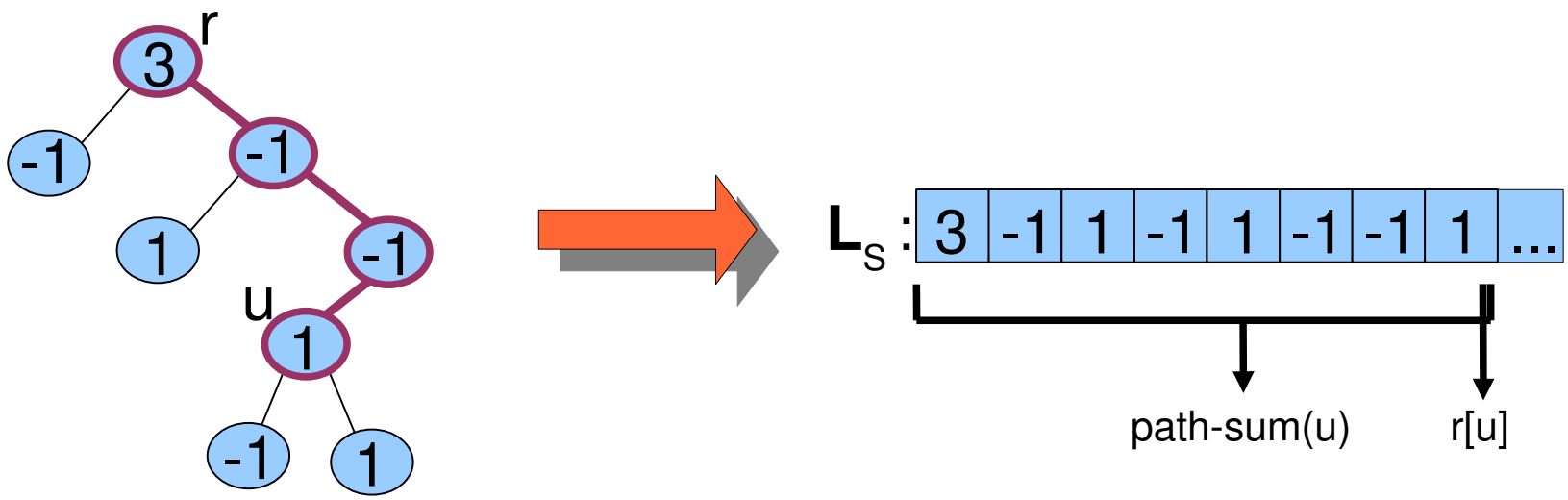
- Define the following labelling $\mathbf{T}_s$ of T:
  - $\mathbf{T}_s(r) = \mathbf{D}[s,r]$
  - $\mathbf{T}_s(u) = \mathbf{D}[s,u] - \mathbf{D}[s,f(u)]$ ($\in \{-1,0,1\}$) for $u \neq r$
- D[s,u] equals to *path-sum* at u in $\mathbf{T}_s$



Encoding **D**[s...]
⇔
Store $\mathbf{T}_s$ with fast path-sum

From path- to prefix-sum

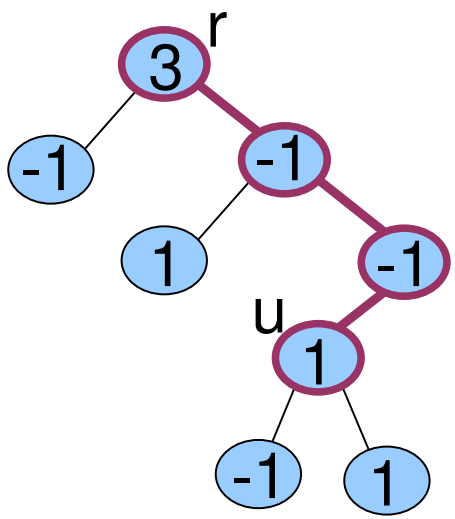
- Solution: “linearize” $\mathbf{T}_s$ in a sequence $\mathbf{L}_s$ of length 2n

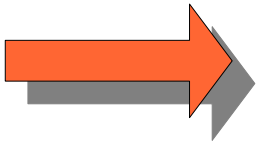


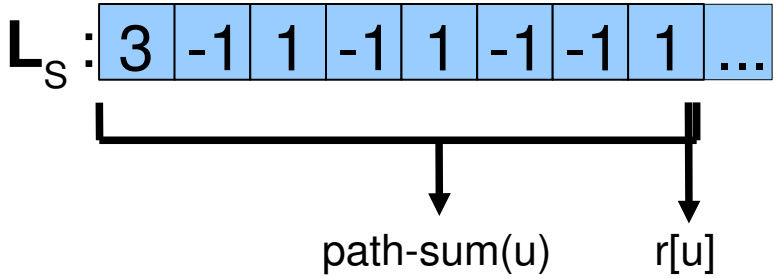
r[ ] = starting times in DFS

From path- to prefix-sum

- Solution: “linearize” $\mathbf{T}_s$ in a sequence $\mathbf{L}_s$ of length 2n





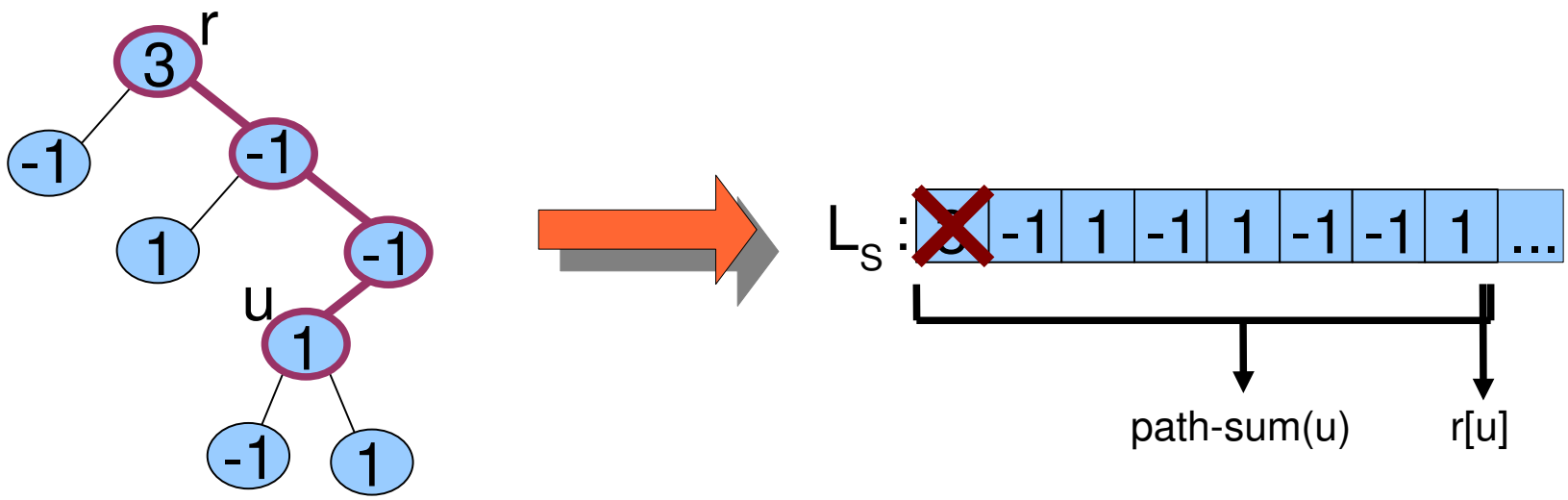


r[ ] = starting times in DFS

Path-sum at node u in $\mathbf{T}_s$ boils down to a prefix-sum at r[u] in $\mathbf{L}_s$

From path- to prefix-sum

- Solution: “linearize” $\mathbf{T}_S$ in a sequence $\mathbf{L}_S$ of length 2n



r[ ] = starting times in DFS

- Build *wavelet tree [Grossi et al., 2003]* on $\mathbf{L}_S$ (root label apart)
  - Space: $2n\gamma + o(n)$ bits, $\gamma = \log_2 3$ (magic constant...)
  - Prefix-sums in O(1) time via *rank* computations

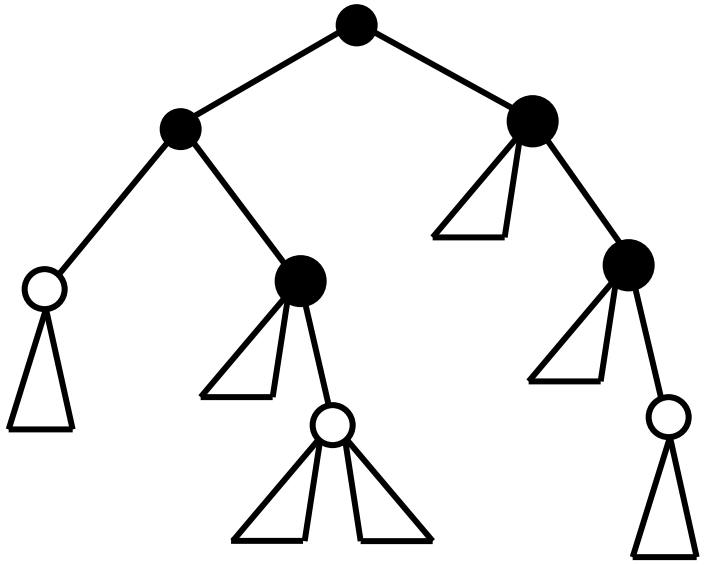
A First Compact Encoding

$\gamma = \log_2 3$

- Encoding of distance matrix **D**[1..n,1..n]:
  - Array **r**[] : O(n log n) bits
  - Prefix-sums on each $\mathbf{L}_s$: $(2\gamma)n + o(n)$ bits
  - Space: $\gamma n^2 + o(n^2)$ bits, Access time: O(1)
- This does not match information-theoretic optimum:
  - γn^2 vs $n^2/2$ bits
  - Linearizations waste space!
    - Need to store two elements per label
- Improved approach: reduce γ to $\gamma/2$
  - Combine above encoding with *macro-micro tree partition*

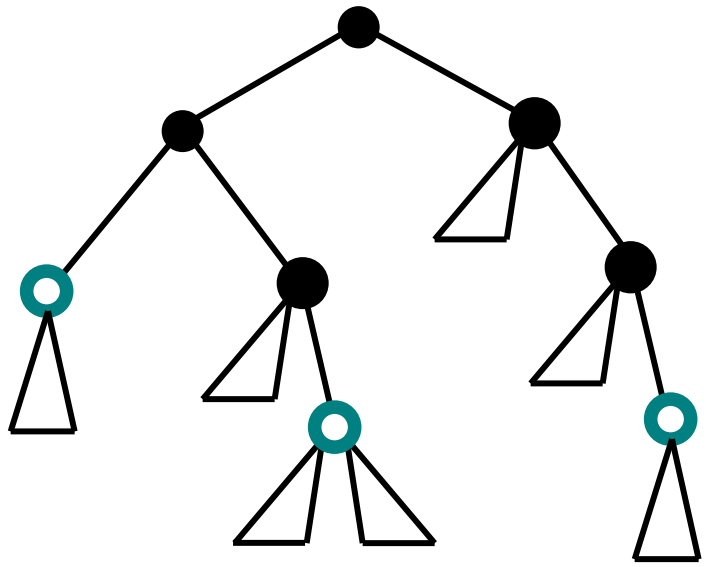
Macro-Micro Tree Partition [Bender et al.,2000]

- Depending on a paramenter α (= c log n, with c < 1), partition the nodes of T into:



Macro-Micro Tree Partition [Bender et al.,2000]

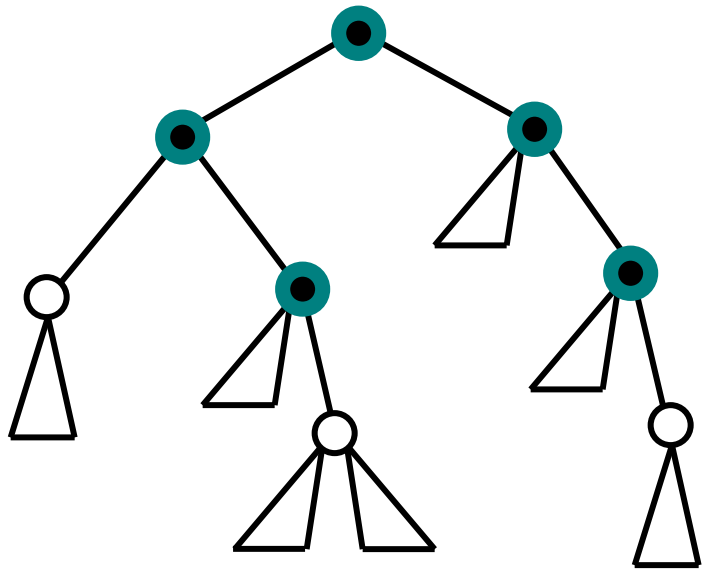
- Depending on a paramenter α (= c log n, with c < 1), partition the nodes of T into:



  - Jump-Nodes: maximally deep nodes having more than α descendants

Macro-Micro Tree Partition [Bender et al.,2000]

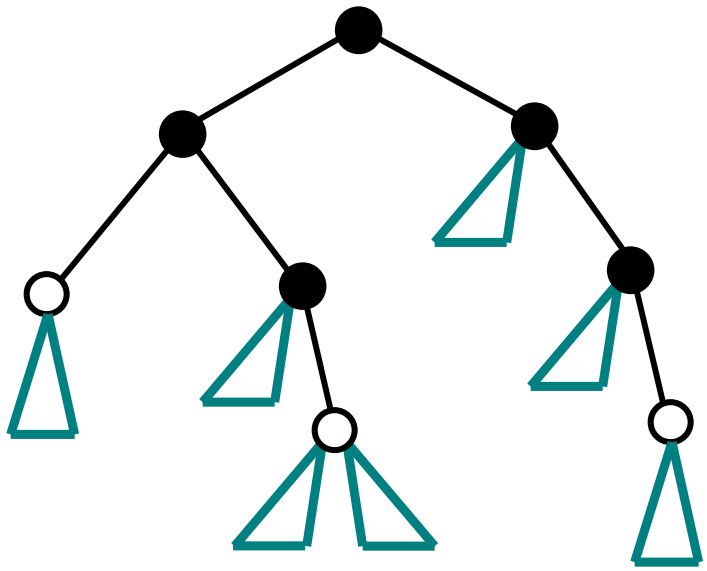
- Depending on a paramenter α (= c log n, with c < 1), partition the nodes of T into:



  - Jump-Nodes: maximally deep nodes having more than α descendants
  - Macro-Nodes: any ancestor of a jump-nodes

Macro-Micro Tree Partition [Bender et al.,2000]

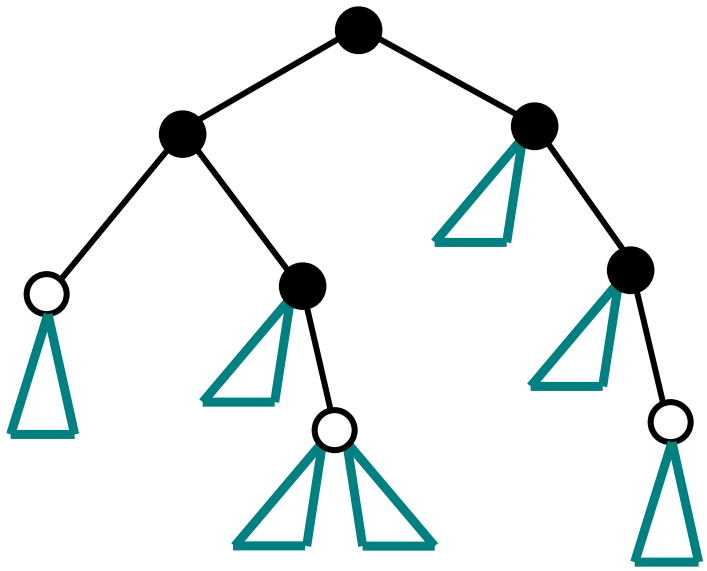
- Depending on a paramenter α (= c log n, with c < 1), partition the nodes of T into:



  - Jump-Nodes: maximally deep nodes having more than α descendants
  - Macro-Nodes: any ancestor of a jump-nodes
  - Micro-Nodes: any other node

Macro-Micro Tree Partition [Bender et al.,2000]

- Depending on a paramenter α (= c log n, with c < 1), partition the nodes of T into:

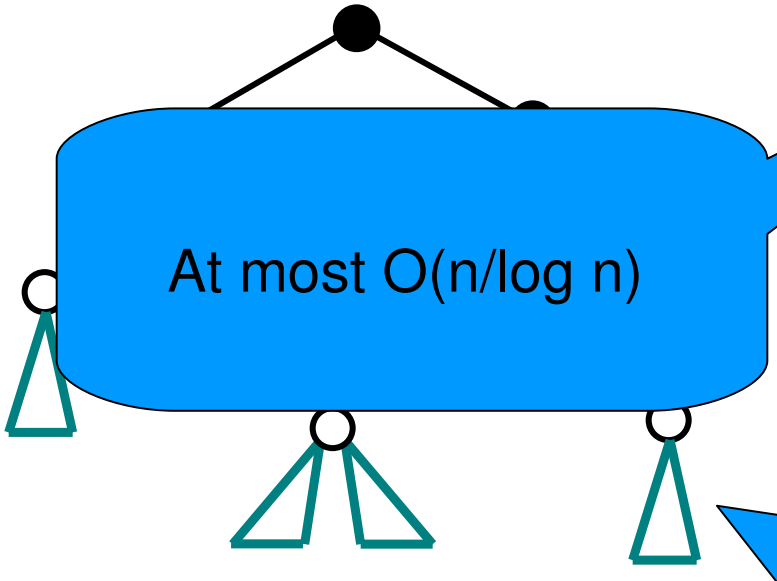


  - Jump-Nodes: maximally deep nodes having more than α descendants
  - Macro-Nodes: any ancestor of a jump-nodes
  - Micro-Nodes: [illegible] node

Micro-tree: maximal connected component of micro-nodes. It contains at most α nodes

Macro-Micro Tree Partition [Bender et al.,2000]

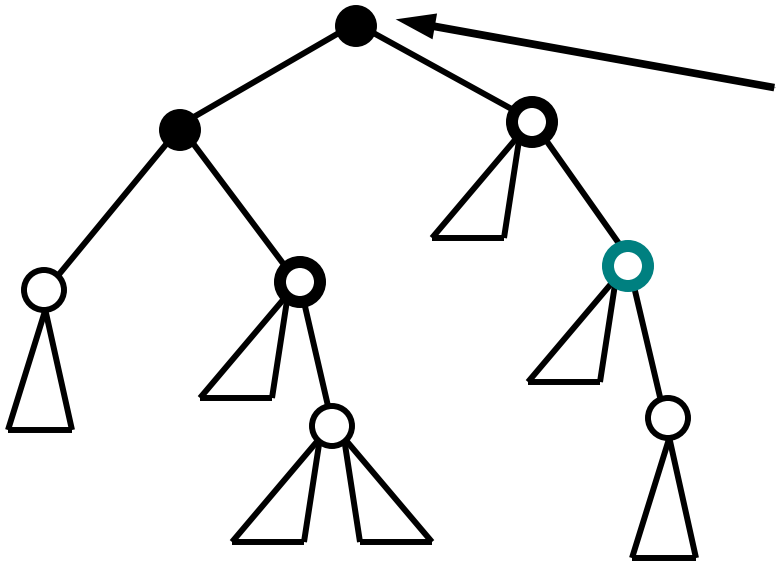
- Depending on a paramenter α (= c log n, with c < 1), partition the nodes of T into:



  - Jump-Nodes: maximally deep nodes having more than α descendants
  - Macro-Nodes: any ancestor of a jump-nodes
  - Micro-tree: maximal connected component of micro-nodes. It contains at most α nodes

Path-Sum on Macro-Micro Partition

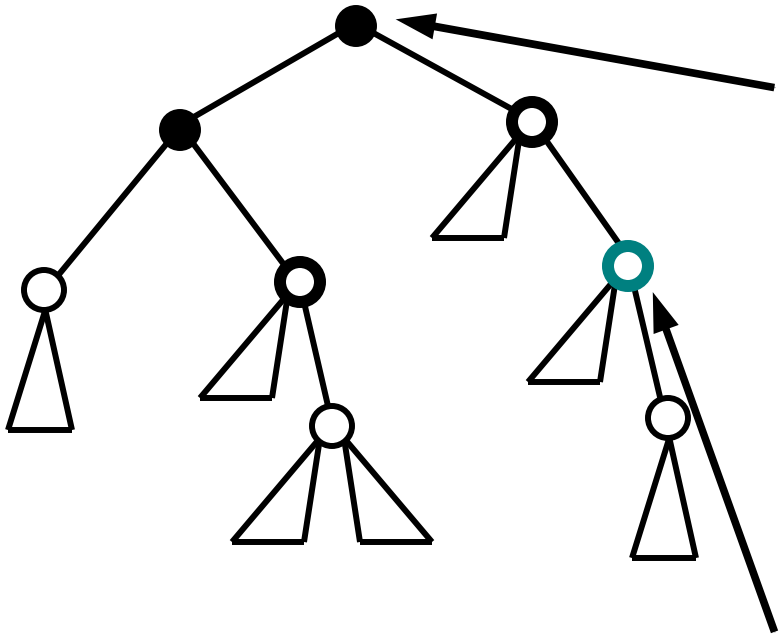
- Some additional definitions:



- Branching-Nodes: more than one macro or jump nodes among children.
- Unary-Nodes: any other macro node

Path-Sum on Macro-Micro Partition

- Some additional definitions:

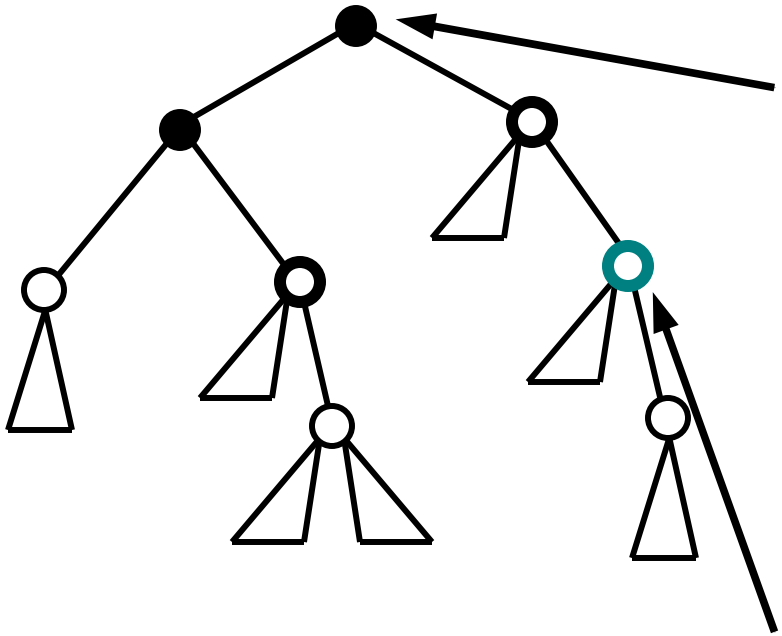


- Branching-Nodes: more than one macro or jump nodes among children.
- Unary-Nodes: any other macro node

In every maximal path of unary nodes, *sample* one out of *log n* nodes.

Path-Sum on Macro-Micro Partition

- Some additional definitions:



At most O(n/log n)

- Branching-Nodes: more than one macro or jump nodes among children.
- Unary-Nodes: any other macro node

At most O(n/log n)

In every maximal path of unary nodes, *sample* one out of *log n* nodes.

Path-Sum on Macro-Micro Partition

- Some additional definitions:

Breaking nodes: the collection of branching, jump and sampled unary nodes.

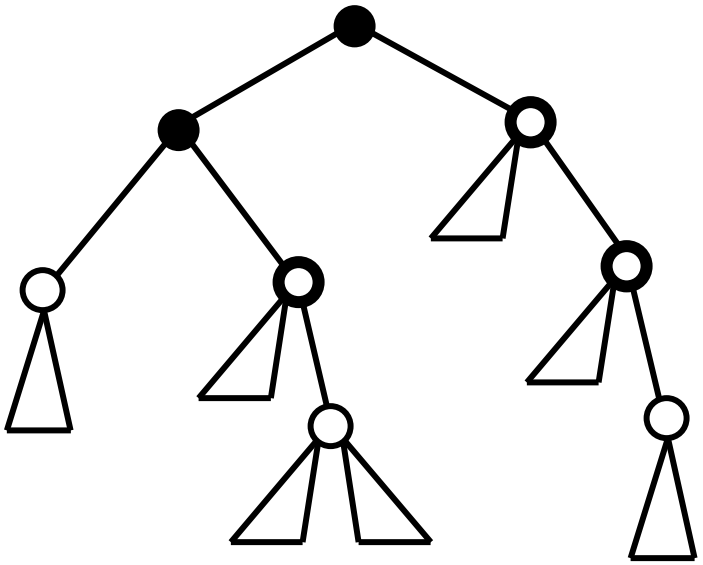
At most O(n/log n)

- Branching-Nodes: more than one macro or jump nodes among children.
- Unary-Nodes: any other macro node

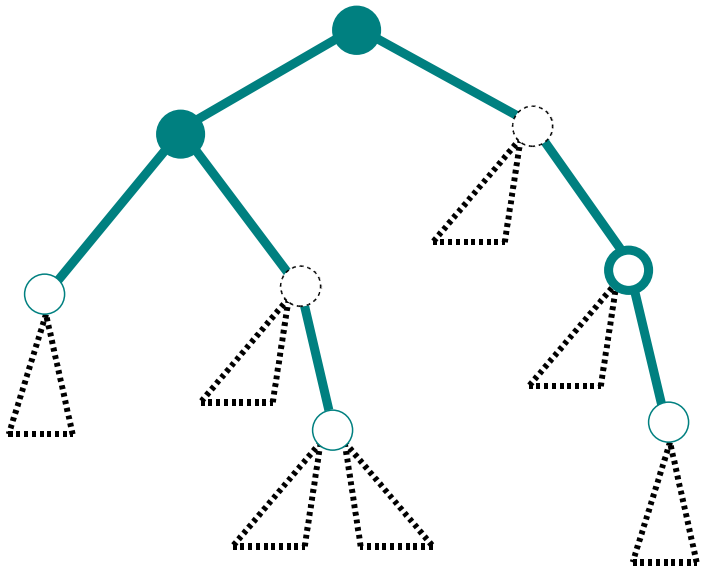
In every maximal path of unary nodes, *sample* one out of *log n* nodes.

At most O(n/log n)

Path-Sums on Breaking Nodes

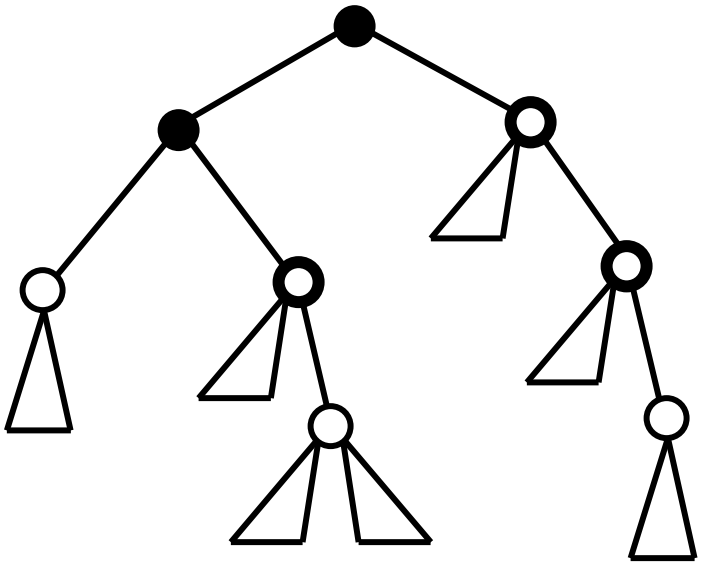


Path-Sums on Breaking Nodes



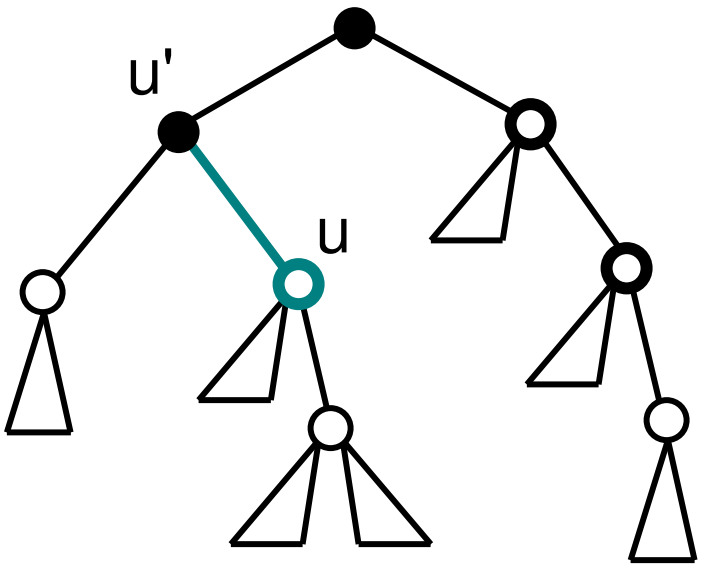
- Build T_F from T by collapsing paths of unary non-breaking nodes
- "Contract" labeling of T onto T_F
- Use a variant of the previous solution to provide path-sums on T_F.
  - Space required : O((n/log n) log log n) = o(n) bits

Path-Sums on Unary Nodes



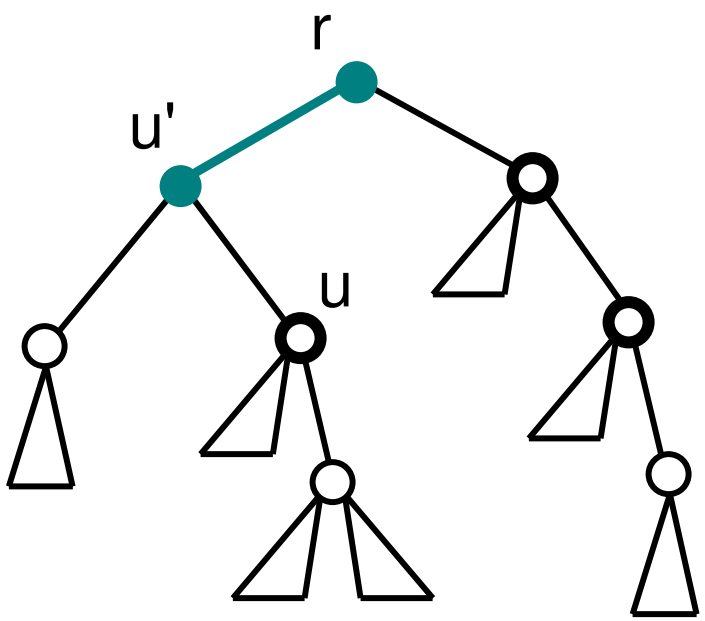
- $\mathbf{M}_T$: sequence of macro-nodes labels sorted according to DF- visit

Path-Sums on Unary Nodes



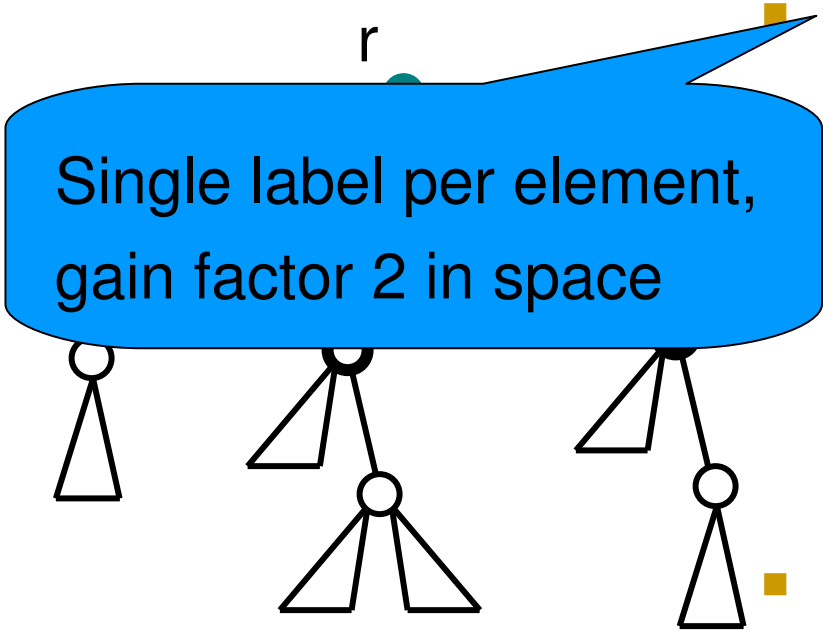
- $\mathbf{M}_T$: sequence of macro-nodes labels sorted according to DF- visit
- To compute path-sum(u):
  - u' : closest breaking ancestor of u
  - Sum in O(1) time labels from u to u'
    - By prefix-sums on $\mathbf{M}_T$

Path-Sums on Unary Nodes



- $\mathbf{M}_T$: sequence of macro-nodes labels sorted according to DF- visit
- To compute path-sum(u):
  - u' : closest breaking ancestor of u
  - Sum in O(1) time labels from u to u'
    - By prefix-sums on $\mathbf{M}_T$
- Add to path-sum(u')

Path-Sums on Unary Nodes

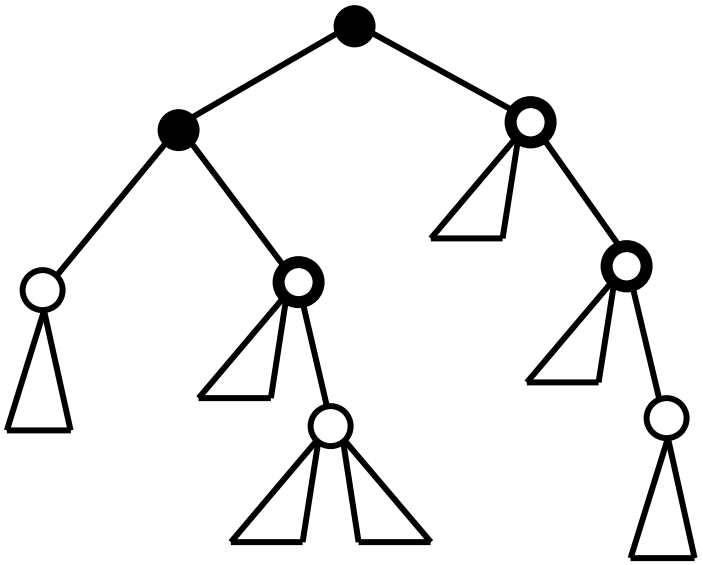


- $\mathbf{M}_T$: sequence of macro-nodes labels sorted according to DF- visit

To compute path-sum(u):

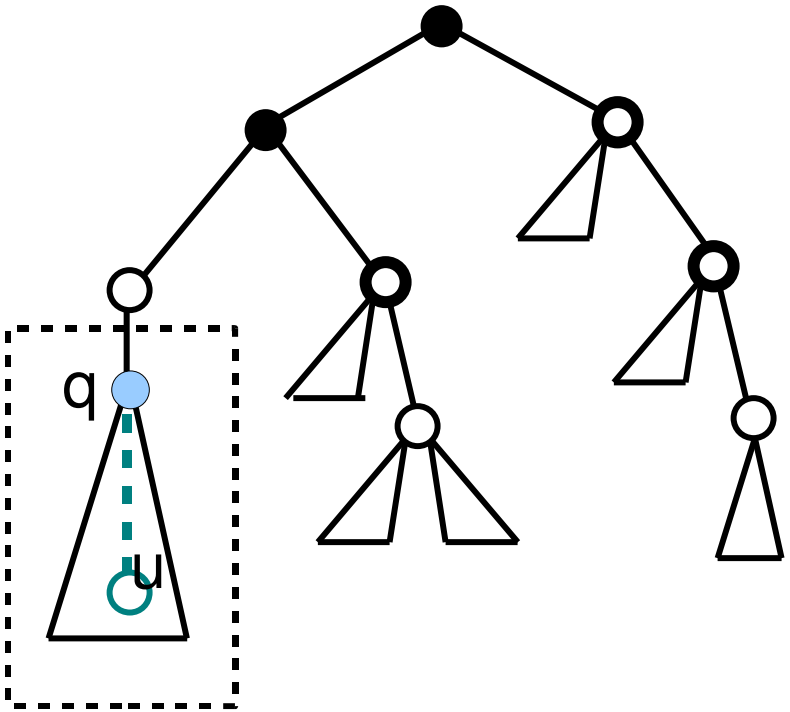
- u' : closest breaking ancestor of u
- Sum in O(1) time labels from u to u'
  - By prefix-sums on $\mathbf{M}_T$
- Add to path-sum(u')

Path-Sums on Micro Nodes



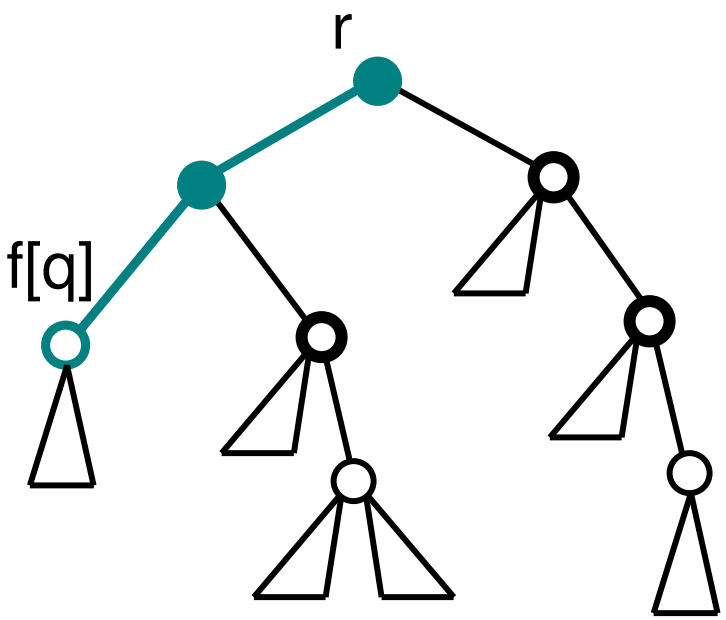
- Tabulate in C all path-sums:
  - for any possible microtree
  - for any possible labeling in {-1,0,1}
- Table C requires o(n) bits
- Store the sequence L of micro-tree labelings
  - O(1) time access to them

Path-Sums on Micro Nodes



- To answer path-sum(u):
  - q: root of micro-tree containing u
  - Sum of labels from u to q in O(1) time using C and L

Path-Sums on Micro Nodes



- To answer path-sum(u):
  - q: root of micro-tree containing u
  - Sum of labels from u to q in O(1) time using C and L
  - Add path-sum(f(q))

Conclusions

- A storage scheme for all-pairs-shortest-path-distance matrices of unweighted graphs:
  - space: $(\log_2 3)/2\ n^2 + o(n^2)$ bits, access time: O(1)
- Open:
  - match the information-theoretic lower bound
  - space dependent on the number of edges
    - Impact on Web and Social Networks applications

Thanks.