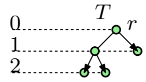
# DFUDS について *Depth First Unary Degree Sequence*




- $E[0]=1,\quad E[2n-1]=0$
- $E[i] = \begin{cases} 1 & D[i]-D[i-1]=+1 \\ 0 & D[i]-D[i-1]=-1 \end{cases}$

平面グラフの強積分解?の解法を動機に
最小共通祖先問題の解法に関する知識が必要になりました

**基本設定:**

- $T=(V,E)$: 特定の頂点 $r$ (根) を持つ (有向) グラフ
- $d: V \to \mathbb{Z}$: 対象の木 $T$ における深さ (根 $r$ までのパスの長さ (辺の個数))
- $\rho: V \to 2^V$: $r$ までのパスに現れる頂点の集まり

区間最小値問題との関連　　$D$: 0, 1, 2, 1, 2, 1, 0, 1, 0

互いに線形時間還元可能!　　$E$ : 1110100100　(((()())())

LCA ← 深さ優先探索に基づく深さ系列 or 括弧表現 → RMQ

RMQ → デカルト木表現 → LCA

*(Cartesian tree)*

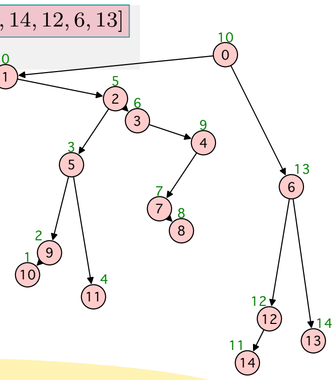
**最小共通祖先問題 (LCA)**

- 前提: 対象となる根付き木 $T$ への前処理が許されている
- 入力: $T$ 内の異なる 2 頂点 $u$ と $v$
- 出力: argmin $\{d(x) \mid x \in \rho(u) \cap \rho(v)\}$

**区間最小値問題 (RMQ)**

- 前提: 与えられた全順序集合の配列 $A$ への前処理が許されている
- 入力: 2つの相異なる索引 $i,\ j \in \binom{|A|}{2},\ i<j$
- 出力: $\mathrm{argmin}_{i \le k \le j} A[k]$

$A := [1,10,9,5,11,2,3,7,8,4,0,14,12,6,13]$
デカルト木の例

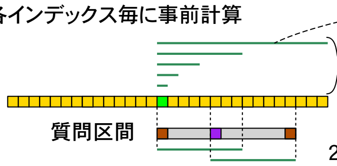


RMQ を効率良く解決するデータ構造について考察する

- 評価基準　⇒　必要領域 & 検索時間
- 領域 $O(n \log n)$ が許されるなら $O(1)$-時間検索は簡単

スパテで! *(sparse table)*

各インデックス毎に事前計算　　区間の長さは $2^\ell$



最大 $\log n$ 個の区間について
最小値およびその位置をメモ

質問区間

2つの重複区間の最小値を解決するだけ!

**配列 $A$ に対応するデカルト木 $T$**

- $T$ は根付きの二分木
- $A \in \mathbb{R}^n$ とするとき
  - $T$ の各頂点は $[n] \times A$ の要素として識別
  - $T$ の根は $(i := \mathrm{argmin} A, \min A)$
  - 根の左の子は $A[0, i-1]$ のデカルト木の根
    右の子は $A[i+1, n-1]$ のデカルト木の根

括弧表現の拡張 DFUDS を用いると
使用領域を $2n + o(n)$ ビット ($\approx O(\log n)$) まで削減して運用できる

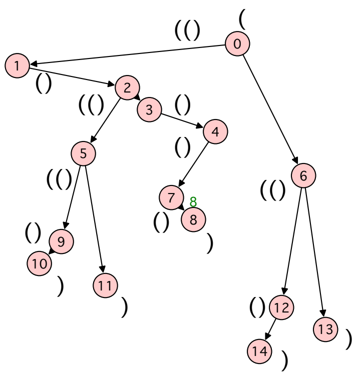
**根付き木から DFUDS への変換**

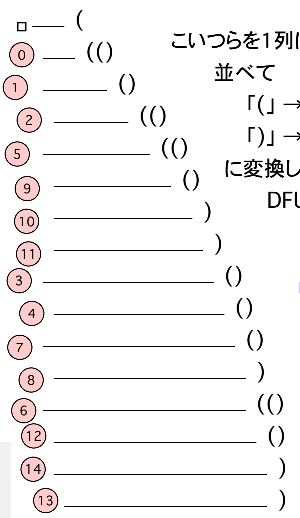
- 深さ優先探索で新頂点の発見時にのみ括弧列が追記される
  - 初期値は開き括弧「(」から
  - 子どもの数だけ開き括弧「(」を追加し、ついでに閉じ括弧「)」を1つ。
    (葉っぱ発見時には閉じ括弧1つぺっとするだけ)


[参考1]　Benoit, Demaine, Munro, Raman, Raman, Rao: (2005).
“Representing trees of higher degree”

[参考2]　Ferrada, Gonzalo: (2017).
“Improved range minimum queries”


この還元括弧列はバランスがとれている
・照明は右の [参考1] にあります





こいつらを1列に
並べて
「(」→ 1
「)」→ 0
に変換したものが
DFUDS!

depth-first ordered
出次数の一進表現 & 深さカウント用ビット
unary degree (rep.)　　　な雰囲気

01-文字列上の演算 (rank と select)

- rank と select
  - $\mathrm{rank}_b(E,i)$ : ビット列 $E$ の先頭から位置 $i$ までの範囲内における文字 $b$ の出現回数
  - $\mathrm{select}_b(E,j)$ : ビット列 $E$ において $j$ 番目の文字 $b$ の位置

※　[参考3] に「$o(n)$ 余剰空間で $O(1)$ 時間解法」の記載があります


[参考3]　D. Clark: (1997). “Compact pat trees.”


これは $O(n)$ 空間使ってしまうのでダメなんだけど…
きっちり解法は存在するので

DFUDS 上のいくつかの演算

- デカルト木の各頂点と閉じ括弧を全単射できるマップを仮想的に持っているとして進める
- DFUDS はバイナリ列と定義しましたけど括弧列としてもアブユーズしていきます
  (入力 $A$ のインデックスや値なんかも付随して参照可能)

- $findclose(i),\ [n] \to [n]$ : 位置 $i$ にある開き括弧に対応する閉じ括弧の位置を返す
- $findopen(i),\ [n] \to [n]$ : 位置 $i$ にある閉じ括弧に対応する開き括弧の位置を返す
- $excess(i),\ [n] \to [n]$ : 位置 $i$ までに出現する開き括弧の個数と閉じ括弧の個数の差
- $enclose(i),\ [n] \to [n]^2$ : 位置 $i$ が開き括弧で、それをきっちり $\underbrace{\text{囲う}}_{\text{直近の}}$ 開閉括弧の位置対