- 书面作业讲解
  - TC第12.1节练习2、5
  - TC第12.2节练习5、8、9
  - TC第12.3节练习5
  - TC第12章问题1
  - TC第13.1节练习5、6、7
  - TC第13.2节练习2
  - TC第13.3节练习1、5
  - TC第13.4节练习1、2、7

# TC第12.1节练习2

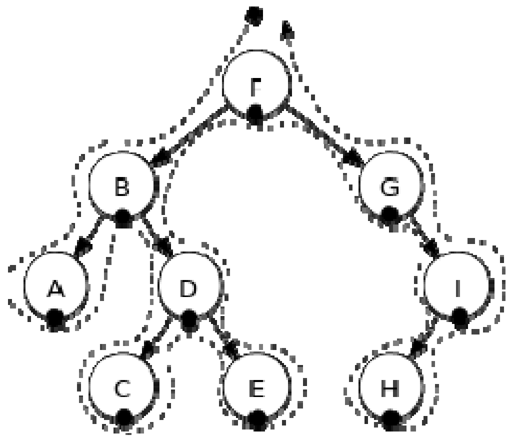
- BST的性质
  - ≥左子节点，≤右子节点，这样对吗？
  - ≥左子树中的节点，≤右子树中的节点

# TC第12.1节练习5

- Any comparison-based algorithm for constructing a binary search tree from an arbitrary list of n elements takes Ω(nlgn) time in the worst case.
  - 反证法：假设只需o(nlgn)，则comparison-based sorting只需o(nlgn)。

# TC第12.2节练习8

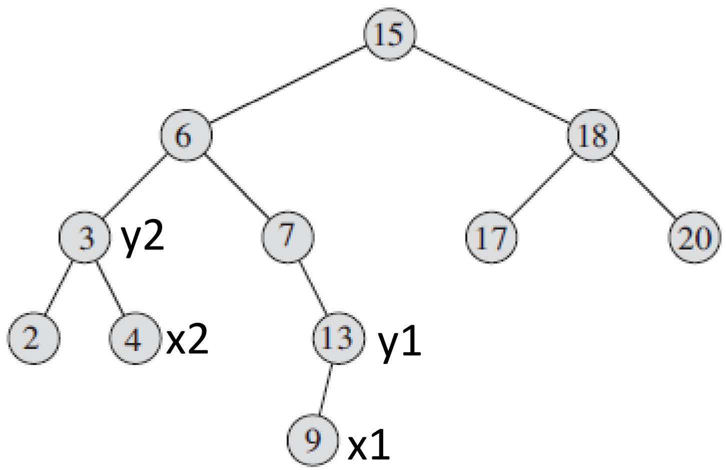
- k successive calls to TREE-SUCCESSOR take O(k+h) time.
  - 其实是在做中序遍历
  - 运行时间即经过顶点的总次数，分两种情况
    - 向上到达
      - 在首顶点向上走到根的路径上（<=h）
      - 其它每次向上到达必然伴随着一次向下到达，不影响渐进时间
    - 向下到达
      - k个后继（=k）
      - 其它都是花费在那些key更大的顶点上，只存在于末顶点到根的路径上（<=h）



# TC第12.2节练习9

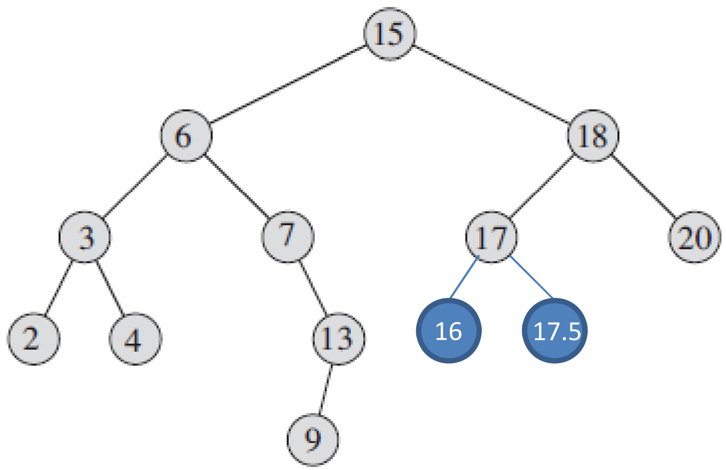
- 为什么y1一定是x1的后继？
- 为什么y2一定是x2的前驱？

- 注意：讨论的范围不能限于以y为根的子树。



# TC第12.3节练习5

- Instead of x.p, keeps x.succ.
  - 实现getParent函数
  - 注意维护受影响顶点的succ

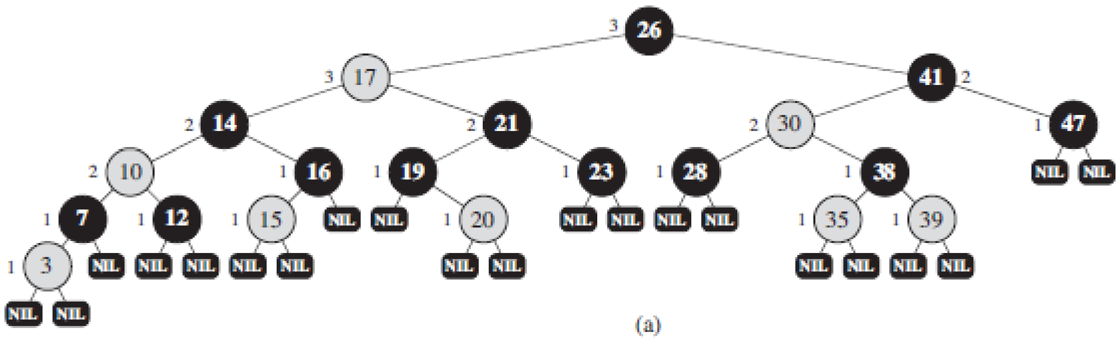


# TC第12章问题1

- (a) insert n items with identical keys.
  - $n^2$
- (b) alternates between x.left and x.right.
  - nlgn
- (c) list
  - n
- (d) randomly between x.left and x.right.
  - Worst-case: $n^2$
  - Expected: nlgn

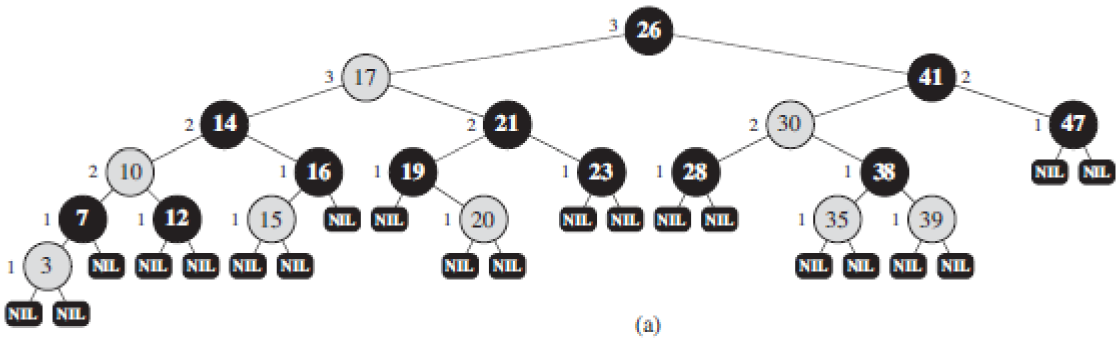
# TC第13.1节练习6

- Number of internal nodes with black-height k?
  - Largest: $2^{2k}$-1，不是$2^{2k+2}$-1（P309: from, but not including, a node…）
  - Smallest: $2^{k}$-1，不是k（ P308: We shall regard these NILs as…）



# TC第13.1节练习7

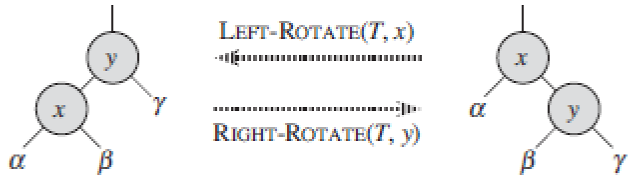
- Ratio of red internal nodes to black internal nodes.
  - Largest: 2
  - Smallest: 0



(a)

# TC第13.2节练习2

- Exactly n-1 possible rotations.
  - 每个rotation都将一个顶点提到了其父顶点的位置
  - 每个非根顶点对应一种被提的rotation，总共n-1种



- 教材答疑和讨论
  - TC第16章第1、2、3节
  - TC第17章

# 问题1：greedy algorithms

- 你怎么向你的师弟师妹们解释greedy algorithm的基本原理？
- 你怎么理解greedy algorithm的两个重要性质？
  - greedy-choice property
  - optimal substructure
- 为什么这两个性质缺一不可？
- 为什么greedy algorithm通常比dynamic programming速度快？

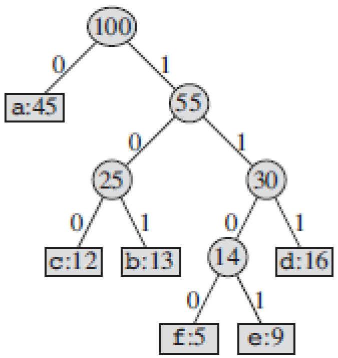
# 问题1：greedy algorithms (续)

- activity-selection problem
  - 你能分别“文科”和“理科”地说明这个问题是什么吗？
  - 采用的greedy algorithm中：
    - greedy choice是什么？
    - 对应的greedy-choice property是什么？
      - 怎么证明？
    - 对应的optimal substructure是什么？
      - 怎么证明？

| $i$ | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $s_i$ | 1 | 3 | 0 | 5 | 3 | 5 | 6 | 8 | 8 | 2 | 12 |
| $f_i$ | 4 | 5 | 6 | 7 | 9 | 9 | 10 | 11 | 12 | 14 | 16 |

# 问题1： greedy algorithms (续)

- Huffman codes problem
  - 你能分别“文科”和“理科”地说明这个问题是什么吗？
  - 采用的greedy algorithm中：
    - greedy choice是什么？
    - 对应的greedy-choice property是什么？
    - 对应的optimal substructure是什么？



# 问题2： amortized analysis

- amortized analysis和average-case analysis有什么异同？
  - per operation vs. per algorithm
  - worst-case vs. average-case

# 问题2： amortized analysis (续)

- 这些问题的分析难在哪儿?
  - stack operations

PUSH$(S, x)$ pushes object $x$ onto stack $S$.

POP$(S)$ pops the top of stack $S$ and returns the popped object. Calling POP on an empty stack generates an error.

```
MULTIPOP(S, k)
    while not STACK-EMPTY(S) and k > 0
        POP(S)
        k = k - 1
```

  - incrementing a binary counter

| Counter value | A[7] | A[6] | A[5] | A[4] | A[3] | A[2] | A[1] | A[0] | Total cost |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 7 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 8 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 10 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 11 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 15 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 16 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 18 |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 19 |
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 22 |
| 13 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 23 |
| 14 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 25 |
| 15 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 26 |
| 16 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 31 |

# 问题2： amortized analysis (续)

- aggregate analysis
  - 基本思路是什么？
  - 如何用来解决这两个问题？

$$\sum_{i=0}^{k-1} \left\lfloor \frac{n}{2^i} \right\rfloor < n \sum_{i=0}^{\infty} \frac{1}{2^i} = 2n ,$$

| Counter value | A[7] | A[6] | A[5] | A[4] | A[3] | A[2] | A[1] | A[0] | Total cost |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 7 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 8 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 10 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 11 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 15 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 16 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 18 |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 19 |
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 22 |
| 13 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 23 |
| 14 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 25 |
| 15 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 26 |
| 16 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 31 |

PUSH$(S, x)$ pushes object $x$ onto stack $S$.

POP$(S)$ pops the top of stack $S$ and returns the popped object. Calling POP on an empty stack generates an error.

```
MULTIPOP(S, k)
while not STACK-EMPTY(S) and k > 0
    POP(S)
    k = k - 1
```

# 问题2： amortized analysis (续)

- accounting method
  - 基本思路是什么？
  - 这个式子是什么意思？为什么这样要求？

$$\sum_{i=1}^{n} \hat{c}_i \geq \sum_{i=1}^{n} c_i$$

  - 如何用来解决这两个问题？
  - 上述要求是如何保证满足的？

PUSH$(S, x)$ pushes object $x$ onto stack $S$.

POP$(S)$ pops the top of stack $S$ and returns the popped object. Calling POP on an empty stack generates an error.

```
MULTIPOP(S, k)
  while not STACK-EMPTY(S) and k > 0
      POP(S)
      k = k − 1
```

| Counter value | A[7] | A[6] | A[5] | A[4] | A[3] | A[2] | A[1] | A[0] | Total cost |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 7 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 8 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 10 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 11 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 15 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 16 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 18 |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 19 |
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 22 |
| 13 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 23 |
| 14 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 25 |
| 15 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 26 |
| 16 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 31 |

# 问题2：amortized analysis (续)

- potential method
  - 基本思路是什么？

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) .$$

$$\sum_{i=1}^{n} \hat{c}_i = \sum_{i=1}^{n} (c_i + \Phi(D_i) - \Phi(D_{i-1})) = \sum_{i=1}^{n} c_i + \Phi(D_n) - \Phi(D_0) .$$

  - 对potential function有什么要求？ $\Phi(D_i) \geq \Phi(D_0)$ for all $i$
  - 如何用来解决这两个问题？

| Counter value | A[7] | A[6] | A[5] | A[4] | A[3] | A[2] | A[1] | A[0] | Total cost |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 3 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 4 |
| 4 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 7 |
| 5 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 8 |
| 6 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 10 |
| 7 | 0 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 11 |
| 8 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 15 |
| 9 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 16 |
| 10 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 0 | 18 |
| 11 | 0 | 0 | 0 | 0 | 1 | 0 | 1 | 1 | 19 |
| 12 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 0 | 22 |
| 13 | 0 | 0 | 0 | 0 | 1 | 1 | 0 | 1 | 23 |
| 14 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 25 |
| 15 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 26 |
| 16 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 31 |

PUSH$(S, x)$ pushes object $x$ onto stack $S$.

POP$(S)$ pops the top of stack $S$ and returns the popped object. Calling POP on an empty stack generates an error.

MULTIPOP$(S, k)$

```
while not STACK-EMPTY(S) and k > 0
    POP(S)
    k = k - 1
```

你对potential function的选择有什么想法？

$$\Phi(D_i) - \Phi(D_{i-1}) \leq (b_{i-1} - t_i + 1) - b_{i-1} = 1 - t_i .$$

$$\hat{c}_i = c_i + \Phi(D_i) - \Phi(D_{i-1}) \leq (t_i + 1) + (1 - t_i) = 2 .$$

# 问题3： dynamic tables

- 为什么会有table expansion问题？
- 连续插入n次，每次的实际代价是多少？

$$c_i = \begin{cases} i & \text{if } i-1 \text{ is an exact power of 2}, \\ 1 & \text{otherwise}. \end{cases}$$

# 问题3： dynamic tables (续)

- aggregate analysis

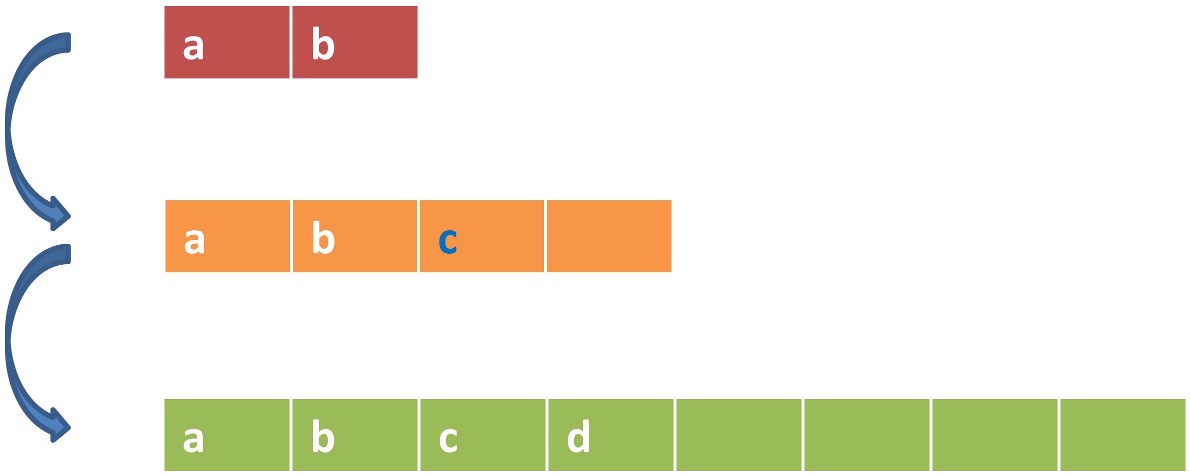
$$c_i = \begin{cases} i & \text{if } i-1 \text{ is an exact power of } 2 , \\ 1 & \text{otherwise} . \end{cases}$$

⇨

$$\begin{aligned} \sum_{i=1}^{n} c_i &\le n + \sum_{j=0}^{\lfloor \lg n \rfloor} 2^j \\ &< n + 2n \\ &= 3n , \end{aligned}$$

# 问题3： dynamic tables (续)

- accounting method
  - 为什么amortized cost是3？以插入c为例说明



# 问题3： dynamic tables (续)

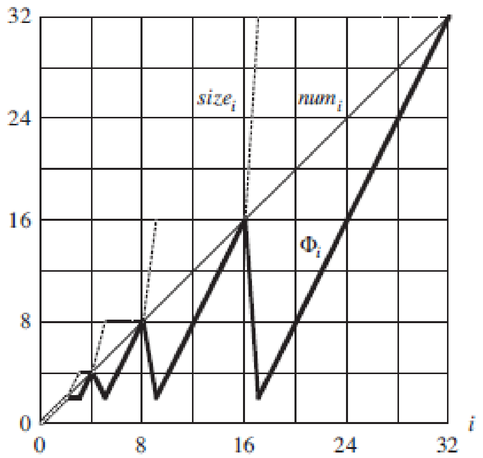
- potential method
  - 怎么想到这样定义potential function的？

$$\Phi(T) = 2 \cdot T.num - T.size$$

  - amortized cost
    - 未发生expansion
    - 发生expansion
  - potential function的走势

$$\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= 1 + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \\
&= 1 + (2 \cdot num_i - size_i) - (2(num_i - 1) - size_i) \\
&= 3 .
\end{aligned}$$

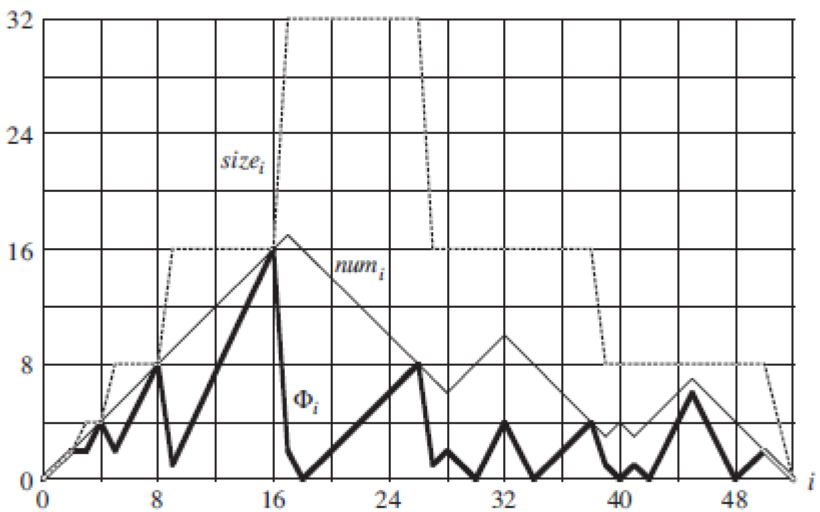
$$\begin{aligned}
\hat{c}_i &= c_i + \Phi_i - \Phi_{i-1} \\
&= num_i + (2 \cdot num_i - size_i) - (2 \cdot num_{i-1} - size_{i-1}) \\
&= num_i + (2 \cdot num_i - 2 \cdot (num_i - 1)) - (2(num_i - 1) - (num_i - 1)) \\
&= num_i + 2 - (num_i - 1) \\
&= 3 .
\end{aligned}$$




能不能结合accounting method
来解释这个走势的含义？

# 问题3： dynamic tables (续)

- 为什么会有table contraction问题？
- 从accounting method的角度考虑：
  - 为什么在load factor=1/2时立即contraction不是好方案？
  - 如何选择最佳的load factor执行contraction？
- potential function的走势




能不能结合accounting method
来解释这个走势的含义？