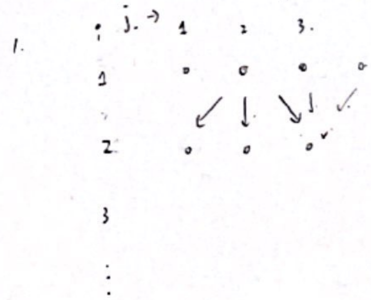
110062619 楊淨富

1.



Optimal substructure:

令 R[i][j] 為 到 B[i][j] 所能取得之"最佳" coins 數.

令 C[i][j] 為 B[i][j] 所擁有之 coin 數.

claim: $\underset{\text{最佳}}{\overset{a}{R[i][j]}} = C[i][j] + \underset{\text{最佳}}{\max}(\overset{b}{R[i-1][j-1]}, \overset{c}{R[i-1][j]}, \overset{d}{R[i-1][j+1]})$.

prove by contradiction,

假設存在 b' 為子問題之最佳解 $(b' > \max(b,c,d))$.

則 $b' + c[i][j] = a' > a$ →← (∵ a 是最佳解).

Algo:

$$\begin{cases} R[i][j] = -\infty, & \text{if } i \le 0 \text{ or } i > n \text{ or } j \le 0 \text{ or } j > n \text{ (Index out of bound)}. \\ R[i][j] = c[i][j] + \max(R[i-1][j-1], R[i-1][j], R[i-1][j+1]), & i = 1 \sim n,\ j = 1 \sim n \end{cases}$$

Base case:

$R[1][j] = C[1][j]$, $j = 1 \sim n$

Inductive case:

```
max = -∞;
for i = 2 ~ n.
    for j = 1 ~ n.
        compute R[i][j] (by above 遞迴式).
        if R[i][j] > max
            max = R[i]
return max
```

Time: 由於表格大小 (R[i][j] 大小) 為 $n \times n$.

且每格皆只須填值

故 $O(n^2)$.

2.

greedy choice:

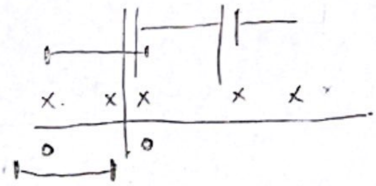
假設 $S = \{x_1, x_2, \dots, x_n\}$. , line segment numbers = LSN

每次取出 S 中最小值 x_i, 且若 $x_{i+1} \le x_i + 1$, 則 x_{i+1} 也一併取出, LSN ++.
$(i+1 \le n)$.

i 的 index 從 1 to n ∴ Time: $O(n)$ ∵ 掃過一輪 S 即可得 LSN.

claim: greedy choice is correct.

prove. by cut and paste.



假設 optimal solution 中
包含 x_1 之 line segment. 令為 L1.

case 1: 若 L1 包含 x_2.

則 $x_2 - x_1 \le 1$.

即拿掉 L1, 改用 greedy choice 之 line segment 亦可包含 x_1, x_2 (∵ $x_2 - x_1 \le 1$).

case 2: 若 L1 不包含 x_2.

↳ • case 2.1. 若包含 x_j, $3 \le j \le n$.

則 $x_j - x_1 \le 1$.

若 $x_1 \sim x_j$ 之間仍存在 x_i ($x_1 < x_i < x_j$).

則 by greedy choice. line segment = $[x_1, x_1+1]$ $[x_j, x_j+1]$
↑ 會包含 x_i.

則拿掉 L1. 改用 greedy choice 仍至少和 OPT 一樣少的 line segments
↳ ∵ OPT 還必須用另一條 line 去包含 x_i.

• case 2.2. 若不包含 x_j, $3 \le j \le n$

則 L1 只有 x_1

則拿掉 L1, 改用 greedy choice 之 line segment: $[x_1, x_1+1]$

仍能和 OPT 一樣.

根據以上 3 cases. 可發現 greedy choice 至少和 OPT 一樣少之 line segments.

故 greedy choice is correct.

OPT substructure:

假設 OPT 是最佳 solution, 且 OPT 含 x_1 之 line 為 L1,

claim: $\overset{\text{"}a}{\text{OPT 之 line set}} = \overset{\text{"}b}{\underset{\text{最佳}}{L1}} + \overset{\text{"}c}{\text{OPT 之 line set} - L1}$.

prove by contradiction.

假設存在 $b' < b$ 使 $b' + c = a' < a$ →← (∵ a 是最佳).

Scanned with CamScanner

3.

By aggregate method.

for a sequence of n operations with INC & 2-Power-INC (x).

累加至 n 時，所需走訪的 bit 較元如下

n	A[7]	A[6]	A[5]	A[4]	A[3]	A[2]	A[1]	A[0]
0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	1
2	0	0	0	0	0	0	1	0
3	0	0	0	0	0	0	1	1
⋮								

假設 $2^{t-1} < n \le 2^t$，其中 $t \ge 1$.

執行 n 次 operations 時，A[0] 需走訪至多 2^t 次

A[1] 〃 2^{t-1} 次

⋮

A[t] 〃 2^0 次

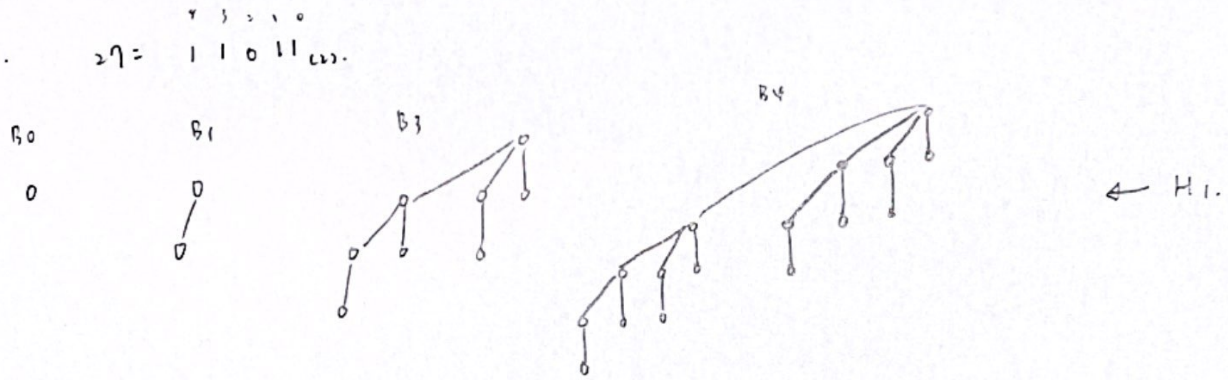
因此總成本為 $T(n) \le 2^t + 2^{t-1} + \dots + 2^0 = 2^{t+1} - 1 = 4 \cdot 2^{t-1} - 1 < 4n - 1 = O(n)$.

所以每次執行 INC / 2-power-INC 之平均分攤成本為 $\frac{T(n)}{n} = \frac{O(n)}{n} = O(1)$.

Scanned with CamScanner

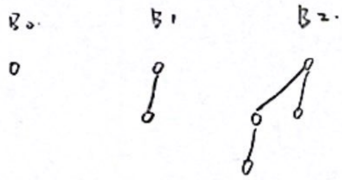
4.

(a). $27 = 11011_{(2)}$.



(b). $7 = 111_{(2)}$.

H_2 →.



referenced from lecture note 17.

(c).
- To Union (H_1, H_2), we process all binomial trees in the two heaps with same order together, starting with "smaller" order first.
- Let k be the order of the set of binomial trees we currently process

3 cases:

1. If there is only one B_k → done
2. If there are two B_k → Merge together, forming B_{k+1}.
3. If there are three B_k → Leave one, merge remaining to B_{k+1}.

After that process next k.

So, Union (H_1, H_2). ⇒ B_5 & B_1.

① $B_0 \times 2$. ⇒. B_1.

② $B_1 \times 3$ ⇒. leave one B_1, get a B_2

③ $B_2 \times 2$ ⇒ get a B_3.

④ $B_3 \times 2$ ⇒ get a B_4.

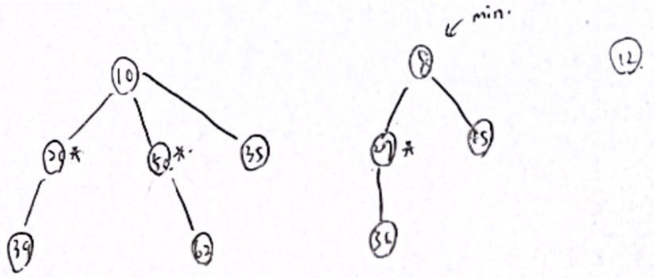
⑤ $B_4 \times 2$ ⇒ get a B_5.

```
   1 1. 1 1.
 1 1 0 1 1
     1 1 1
-----------
1 0 0 0 1 0
↓       ↓
B5      B1.
```

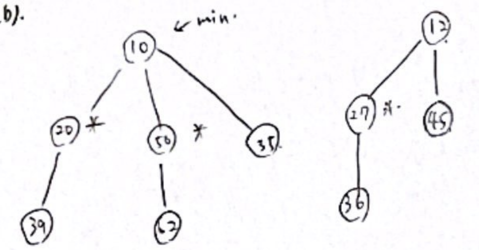
Scanned with CamScanner

5.

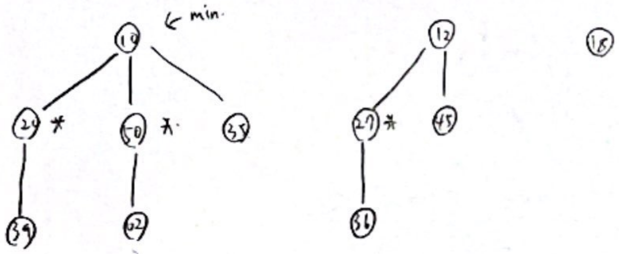
(a).



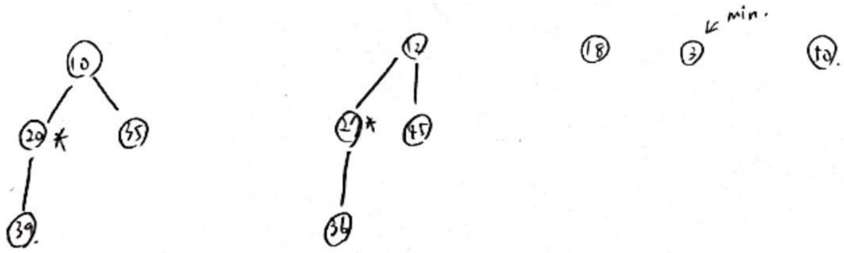
(b).



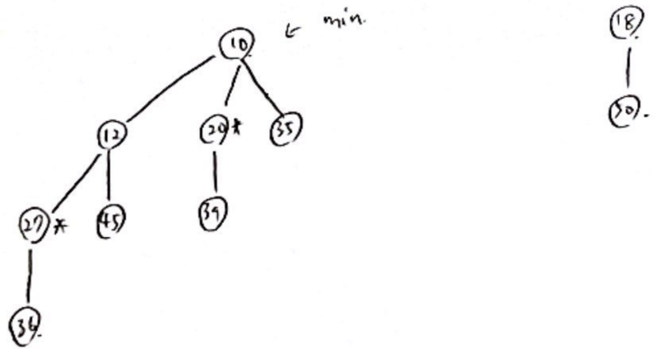
(c).



(d).



(e).



6.

令 $Best_0[v]$ = min. cost we can get at subtree rooted at v with no store at v.

$Best_1[v]$ = 〃 with a store at v.

$$Best_0[v] = \min_{u \in \cancel{\text{neighbor}}\ \text{child}(v)} \{ Best_1[u] \}$$

$$Best_1[v] = cost[v] + \sum_{u \in \cancel{\text{neighbor}}\ \text{child}(v)} Best[u] \quad (Best[u] = \min(Best_0[u], Best_1[u])).$$

Base case:

leaf node$_i$. $\begin{cases} Best_0[leaf_i] = 0 \\ Best_1[leaf_i] = cost[leaf_i] \end{cases}$

Inductive case:

從 leaf 的 parent 開始 朝 root，往上計算 每個 node v 之 $Best_0[v]$, $Best_1[v]$.

最後 return $\min(Best_0[root], Best_1[root])$.

Time:

Each $Best[v]$ can be computed in $O(1)$ time.

$\Rightarrow$ Running time is $O(n)$.

OPT substructure:

OPT sol:

$\underbrace{\min(Best_0[root], Best_1[root])}_{a}$ = $\underbrace{\text{子問題 1}}_{b}$ + $\underbrace{\text{子問題 2}}_{c}$ (with above recursive relationship).

prove by contradiction.

假設子問題存在最佳解 $b' < b$

$\rightarrow$ $b' + c = a' < a$ $\rightarrow\leftarrow$ ($\because$ a 是最佳解).