H30 [1]

(1) 非負自然数 k は高々 $\log(n+1)$ この bit を用いて2進数表記できる。
また、含まれる1の個数が k この要素は ${}_nC_k$ 通りしか存在しないため、これらにそれぞれ $0 \sim {}_nC_k - 1$ までの相異なるインデックスを割り合てれば $\log {}_nC_k$ この bit を用いて2進法表記できる。
以上2つの情報をあわせて、Z を k も含めて符号化するための符号長は高々

$$\log(n+1) + \log {}_nC_k \quad (\text{bit}) \text{ で十分}$$ ■

(2) 題意より

$$f(\theta) = \sum_{t:x_t=1} (y(t)-\theta)^2$$

$$= \sum_{t\in\{1,2,7,8\}} (y(t)-\theta)^2 = 3(1-\theta)^2 + (0-\theta)^2 = 4\theta^2 - 6\theta + 3$$

$$= 4\left(\theta - \frac{3}{4}\right)^2 + \frac{3}{4} \quad \text{なので} \quad \underline{\theta = \frac{3}{4}}$$

(3) $L(Z) = \log(n+1) + \log {}_nC_k$ (但し、k は Z に含まれる bit 数)

・i=1 $\left(y_0^{(1)}, y_1^{(1)}\right) = (1000, 1110)$ より

$$\Delta(1|y) = L(1000) + L(1110)$$
$$= \log 5 + \log 4 + \log 5 + \log 4 = \log 400$$

・i=2 $\left(y_0^{(2)}, y_1^{(2)}\right) = (11000, 110)$ より

$$\Delta(2|y) = L(11000) + L(110)$$
$$= \log 6 + \log 10 + \log 4 + \log 3 = \log 720$$

・i=3 $\left(y_0^{(3)}, y_1^{(3)}\right) = (110, 11000)$ より

$$\Delta(3|y) = L(110) + L(11000)$$
$$= \log 4 + \log 3 + \log 6 + \log 10 = \log 720$$

・$i=4$ $\left(y_0^{(4)},\ y_1^{(4)}\right) = \left(1100,\ 1100\right)$ より

$$\Delta(i|y) = L(1100) + L(1100)$$
$$= \log 5 + \log 6 + \log 5 + \log 6 = \log 900$$

以上より $\Delta(i|y)$ を最小にする i は $i=1$, $i=4$ である. $\left(\Delta(1|y) < \Delta(2|y) = \Delta(3|y) < \Delta(4|y)\right)$

(4) $i^*=1$ により S を $y_1^{(1)}$, $y_1^{(0)}$ に分割した後の各ブロックの $(x_2, x_3, x_4), y$ のデータをまとめると以下になる

$y_1^{(1)}$

t	x_2	x_3	x_4	y
1	0	0	1	1
2	1	0	0	1
3	0	1	1	1
8	1	1	0	0

$y_1^{(0)}$

t	x_2	x_3	x_4	y
4	1	1	0	1
5	0	1	1	0
6	0	1	0	0
7	0	0	1	0

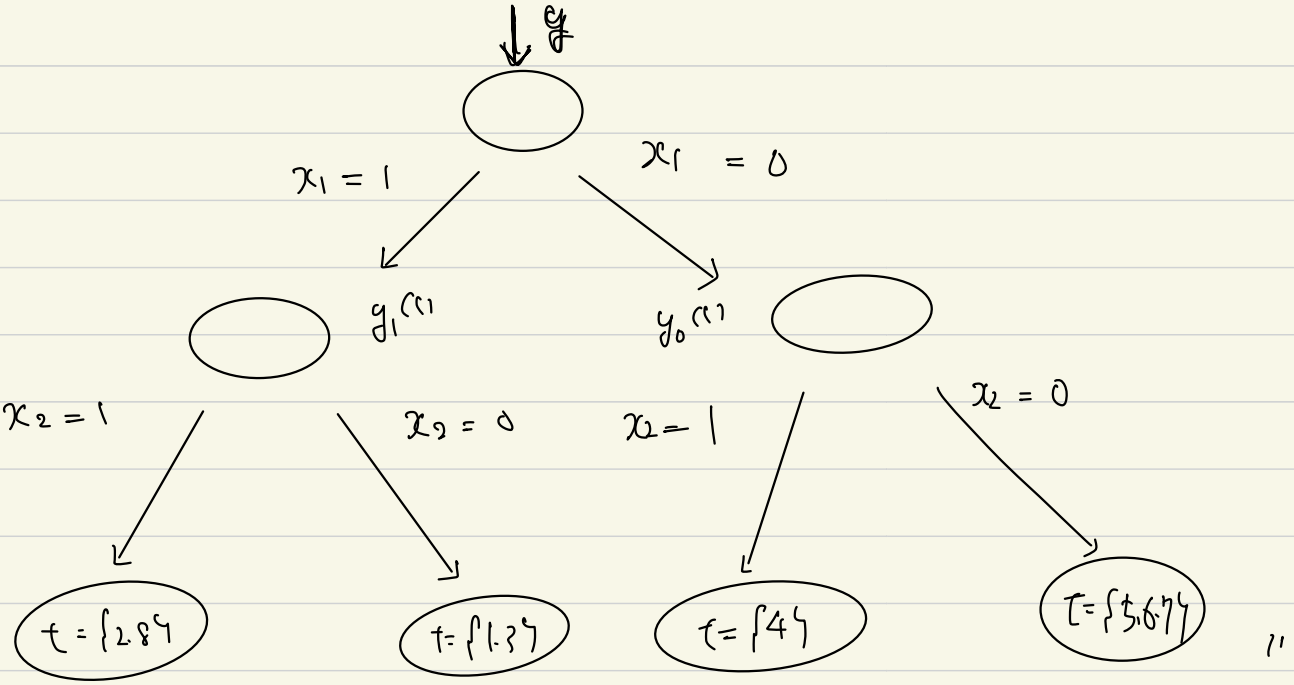
同様に $i=2, 3, 4$ について $\Delta(i|y_1^{(1)})$, $\Delta(i|y_0^{(1)})$ を計算すると以下のようになる.

$$\begin{cases} \Delta(2|y_1^{(1)}) = L(10) + L(11) = \log 18 \\ \Delta(3|y_1^{(1)}) = L(10) + L(11) = \log 18 \\ \Delta(4|y_1^{(1)}) = L(11) + L(10) = \log 18 \end{cases}, \quad \begin{cases} \Delta(2|y_0^{(1)}) = L(1) + L(000) = \log 8 \\ \Delta(3|y_0^{(1)}) = L(100) + L(0) = \log 24 \\ \Delta(4|y_0^{(1)}) = L(00) + 2(10) = \log 18 \end{cases}$$

したがって. $i=2$ (ランダム) , $i=2$

をそれぞれ次の分割に用いる.

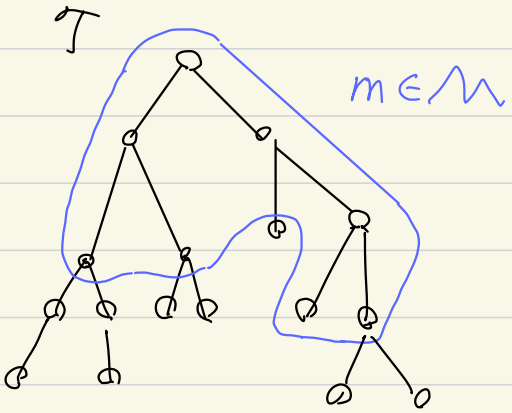
葉の深さが2になった時点で分割は終了するので、これで終了. 得られる分割木は次ページのようになる.



(5) 機械学習分野における，いわゆる「過学習」とよばれる訓練データに過度に適合したモデルの作成が原因で，より一般的なデータにあてはまるであろう特徴量を見逃してしまうという現象が発生するためである．

この場合，モデル用訓練データは S であり，テストデータは新たに与えられるデータとなる．過学習を防ぎ，一般的なモデルに対する予測率を上げるためには，純粋に訓練データの数を増やすか，訓練データをいくつかの分割したグループに分け，それらの部分和でのモデルを合わせて全体のモデルを構築するといった方法が考えられる．

(6) 分割木での頂点数を N とした時，$O(N^2)$ で動作するアルゴリズムを考える．但し，あらかじめもとの木において，各頂点に分類される木の番号一覧を各頂点に保存してゆくものとする．…(*)



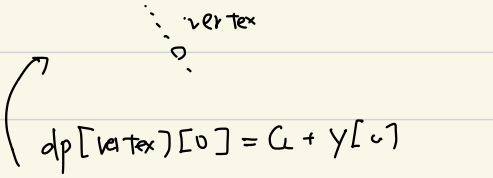
(もとの木で集合 $\mathcal{M}$ に属する部分木 $m \in \mathcal{M}$ の例)

この時，各頂点において，以下の情報を保存する配列を用意する

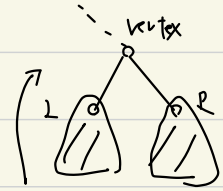
dp[vertex][0/1] := 0/1 vertexから出る2分木の分割を
- 0 … 考慮しない
- 1 … 考慮する

隣の vertex を根とする部分木 M_{vertex} の最小コスト

図として表記すると、



ここまでで vertex に分類された v の集合に対する
L[y] を考える



さらなる分岐を行う.

$dp[vertex][1] = \min_{} dp[Lchild][0/1] + \min_{} dp[Rchild][0/1] + C_I$

これより、dp[vertex][0/1] の値はその子頂点から動的計画法を用いて計算することができる.
したがって、根を始点とする DFS (深さ優先探索) を行い、その帰りがけに dp[vertex][0/1]
の値を更新してゆくことで、任意の頂点に対する dp[vertex][0/1] の値を計算できる
以下にその疑似コードを示す (なお // (コメントアウト) を用いて適宜説明を加えている)

```
void dfs (int u){
    for v ∈ child[u]
        dfs(v)
    // これまでのデータから dp[u] の更新

    dp[u][0] = C_L + C_I + L[y_u]
    // y_u := u を根とする部分木の葉にあつまた全てのデータの y を並べて得られる2元列

    dp[u][1] = C_I + min_{0≤i≤1} dp[leftchild][i] + min_{0≤i≤1} dp[rightchild][i]

    // u の部分木に対する問題をその子子孫の部分木の問題に分割する
}
dfs(root);
```

この時、もともとの問題である $m \in M$ に対する最小ペナルティの値は
$\min_{0 \le i \le 1} dp[root][i]$ に等しいことは明らかである. また、その部分木の復元も
根から順に各頂点をたどってゆき、dp[vertex][0] / dp[vertex][1] のどちらが最適解さえ与えるかを求めることで O(N) で復元可能である.

最後に，前に述べた擬似コードの計算量が$O(N^2)$であることを示そう．

dp[u][0]の更新には，y_uの列挙のコストがかかり，これは$y'_{leftchild}+y'_{rightchild}$
+（自身が葉の場合，y_u）において求められる．この表は高々$O(N)$であるため，取得及び
L[y_u]の計算にも$O(N)$のコストがかかる．

dp[u][1]の更新には，$O(1)$の算述演算のみを要する．

任意の頂点について上記が成り立ち，かつdfs自体も$O(N)$で終了するので，
全体の計算量は高々$O(N^2)$である．