فرادرس
فراتر از یک کلاس درس
www.faradars.org

طراحی الگوریتم

درس پنجم: روش حریصانه

مدرس:

فرشید شیرافکن

دانشجوی **دکتری دانشگاه تهران**

(کارشناسی و کارشناسی ارشد : کامپیوتر نرم افزار) (دکتری: بیو انفورماتیک)

روش حریصانه

الگوریتم حریصانه با انجام یک سری انتخاب، که در جای خود بهینه است، عمل کرده، به امید اینکه یک حل بهینه کلی یافت شود.

در الگوریتم حریصانه، همواره جواب بهینه حاصل نمی شود.

بهینه بودن الگوریتم باید تعیین شود.

در روش حریصانه، تقسیم به نمونه های کوچکتر صورت نمی پذیرد.

فرادرس

نحوه کار الگوریتم حریصانه

الگوریتم حریصانه، کار را با یک مجموعه تهی آغاز کرده و به ترتیب عناصری به مجموعه اضافه می کند تا این مجموعه حلی برای نمونه ای از یک مسئله را نشان دهد.

هر تکرار شامل مولفه های زیر است:

روال انتخاب	عنصر بعدی را که باید به مجموعه اضافه شود، انتخاب می کند. انتخاب طبق یک ملاک حریصانه انجام شده که یک شرط بهینه را در همان برهه برآورده می سازد.
بررسی امکان سنجی	تعیین می کند که آیا مجموعه جدید برای رسیدن به حل، عملی است یا خیر.
بررسی راه حل	تعیین می کند که آیا مجموعه جدید، حل نمونه را ارائه می کند یا خیر.

الگوریتم های حریصانه که بررسی خواهند شد :

۱- خرد کردن پول

۲- زمانبندی

۳- کد هافمن

۴- کوله پشتی کسری

۵- دایکسترا

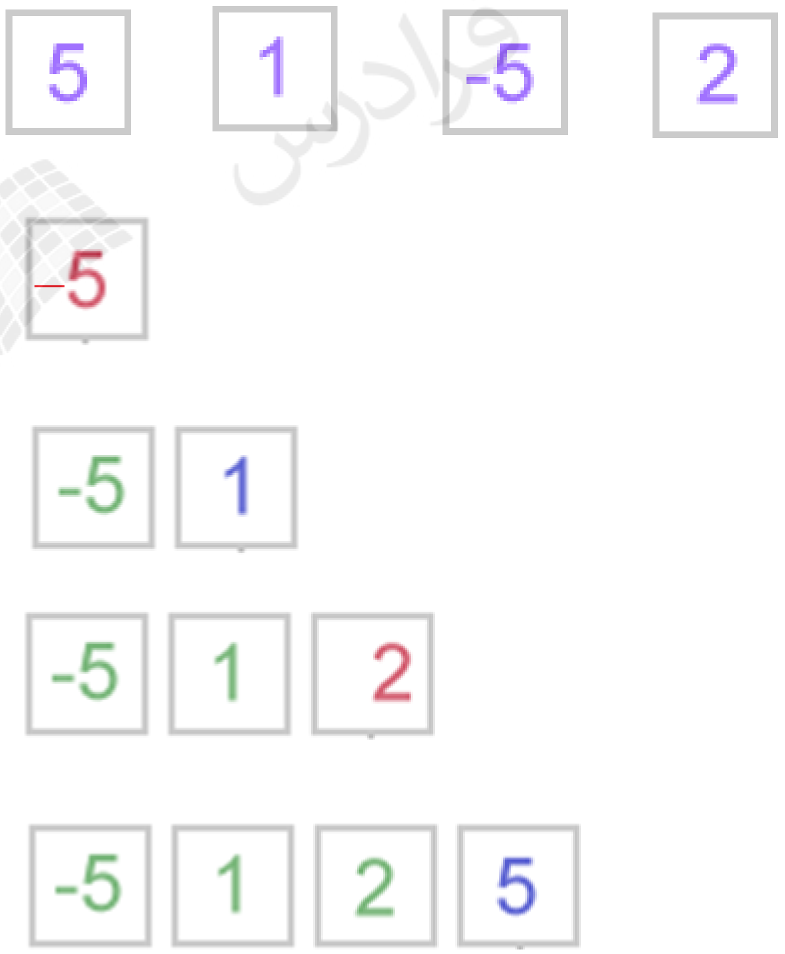
۶- پریم و کروسکال

فرادرس

روش کلی الگوریتم های حریصانه

مرتب سازی انتخابی

$$
\begin{array}{l}
Greedy-Algorithm\,(Set\ C)\{\\
\quad S \leftarrow \emptyset;\\
\quad While\,(not\ Solution(S)\ and\ C \neq \emptyset)\{\\
\qquad x \leftarrow Select(C);\\
\qquad C \leftarrow C - \{x\};\\
\qquad if\,(Feasible(S \cup \{x\}))\{\\
\qquad\quad S \leftarrow S \cup \{x\};\\
\qquad \}\\
\quad \}\\
\quad if\,(Solution(S))\ return(S);\\
\quad else\ return("\,nosolution\,");\\
\}
\end{array}
$$



مسئله خرد کردن پول

هدف فروشنده یک فروشگاه دادن بقیه پول به میزان صحیح و با حداقل تعداد سکه ممکن است. ابتدا فروشنده بزرگترین سکه از لحاظ ارزش را پیدا می کند، یعنی ملاک وی ارزش سکه است(روال انتخاب). سپس باید ببینند که آیا با افزودن این سکه به بقیه پول، جمع کل آنها از چیزی که باید باشد بیشتر می شود یا خیر(امکان سنجی). اگر با افزودن این سکه، بقیه پول از میزان لازم بیشتر نشود، این سکه به مجموعه افزوده می شود.

سپس تحقیق می کند تا ببیند که آیا مقدار بقیه پول با میزان لازم برابر شده یا خیر(**بررسی راه حل**). **اگر برابر نبود، با استفاده از روال انتخاب یک سکه دیگر انتخاب می کند و فرایند تکرار می شود.**

او چندین بار این کار را انجام می دهد که مقدار بقیه پول با میزان لازم برابر شود یا اینکه دیگر سکه ای برایش باقی نماند که در این حالت قادر به بازگرداندن مقدار بقیه پول نیست.

مثال:

اگر سکه های آمریکایی(۱ سنتی، ۵سنتی، ۱۰سنتی، ۲۵سنتی و نیم دلاری) باشند و اگر از هر کدام حداقل یک عدد موجود باشد، آیا الگوریتم حریصانه همواره در صورت وجود حل بهینه را برمی گرداند؟

بله– به طور نمونه برگرداندن 36 سنت به مشتری:

۱– دادن یک سکه 25 سنتی

۲– دادن یک سکه 10 سنتی

۳– دادن یک سکه 10 سنتی دیگر (ولی چون از 36 سنت بیشتر می شود، این سکه پس گرفته می شود.)

۴– دادن یک سکه 5 سنتی (ولی چون از 36 سنت بیشتر می شود، این سکه پس گرفته می شود.)

۵– دادن یک سکه 1 سنتی

بنابراین در کل یک سکه 25 سنتی، یک سکه 10 سنتی و یک سکه 1 سنتی به مشتری داده شد.

فرادرس

مثال:

با توجه به مثال قبل، اگر یک سکه 12 سنتی را جزء سکه های آمریکایی در نظر بگیریم، آیا الگوریتم حریصانه همواره به حل بهینه می انجامد؟

(۱ سنتی، ۵ سنتی، ۱۰ سنتی، ۱۲ سنتی، ۲۵ سنتی و نیم دلاری)

خیر– به طور نمونه برگرداندن 16 سنت :

۱- دادن یک سکه 12 سنتی

۲- دادن یک سکه 10 سنتی دیگر (ولی چون از 16 سنت بیشتر می شود، این سکه پس گرفته می شود.)

۳- دادن یک سکه 5 سنتی (ولی چون از 16 سنت بیشتر می شود، این سکه پس گرفته می شود.)

۴- دادن چهار سکه 1 سنتی

حل حریصانه : پنج سکه (یک سکه ۱۲ سنتی و چهار سکه ۱ سنتی)

حل بهینه : سه سکه (یک سکه ۱۰ سنتی،یک سکه ۵ سنتی و یک سکه ۱ سنتی)

فرادرس

زمان بندی ساده

تعیین زمان بندی بهینه برای سه کار با زمان سرویس های 5، 10 و 4 :

زمان بندی	زمان کل در سیستم
[1,2,3]	5 + (5 + 10) + (5 + 10 + 4) = 39
[1,3,2]	5 + (5 + 4) + (5 + 4 + 10) = 33
[2,1,3]	10 + (10 + 5) + (10 + 5 + 4) = 44
[2,3,1]	10 + (10 + 4) + (10 + 4 + 5) = 43
[3,1,2]	4 + (4 + 5) + (4 + 5 + 10) = 32
[3,2,1]	4 + (4 + 10) + (4 + 10 + 5) = 37

بنابراین برای پیدا کردن زمانبندی بهینه، کارها را از کوچک به بزرگ انجام می دهیم.

فرادرس

- زمان الگوریتمی که همه زمان بندی های ممکن را در نظر می گیرد، به صورت فاکتوریل است.

- پیچیدگی زمانی الگوریتم زمان بندی برابر $W(n) \in \theta(n \lg n)$ است. چون تنها کاری که انجام می دهد مرتب سازی کارها بر حسب زمان سرویس است.

- **تنها زمان بندیی که زمان کل در سیستم را کمینه سازی می کند، زمان بندیی است که در آن کارها بر حسب افزایش زمان ارائه خدمات مرتب می شوند.**

زمان بندی با مهلت معین

در مسئله زمان بندی با مهلت معین، هر کاری در یک واحد زمانی به پایان می رسد و دارای یک مهلت و سود معین است.

اگر هر کار قبل از مهلت معین یا در آن مدت انجام شود، سود مورد نظر حاصل می شود.

هدف، زمان بندی کارها به نحوی است که بیشترین سود حاصل شود.

در این مسئله لازم نیست همه کارها زمان بندی شوند .

یک روش حریصانه منطقی برای حل این مسئله، عبارت از مرتب سازی غیر نزولی کارها بر اساس سود آن ها و سپس وارسی هر یک از کارها به ترتیب و در صورت امکان، افزودن آن به زمان بندی است.

مثال: تعیین زمان بندی با سود بهینه :

کار	مهلت	سود
1	2	30
2	1	35
3	2	25
4	1	40

زمان بندی ها و سودهای ممکن:

زمان بندی	سود کل
[1,3]	30 + 25 = 55
[2,1]	35 + 30 = 65
[2,3]	35 + 25 = 60
[3,1]	25 + 30 = 55
[4,1]	40 + 30 = 70
[4,3]	40 + 25 = 65

[4,1] یک ترتیب امکان پذیر است ولی [1,4] ترتیب امکان پذیری نیست.

فرادرس

مثال:

تعیین یک ترتیب امکان پذیر :

کار	1	2	3	4	5	6	7
مهلت	3	1	1	3	1	3	2
سود	40	35	30	25	20	15	10

۱– S برابر با $\varnothing$ قرار داده می شود.

۲– S برابر با $\{1\}$ قرار داده می شود، چون ترتیب $[1]$ امکان پذیر است.

۳– S برابر با $\{1,2\}$ قرار داده می شود، چون ترتیب $[2,1]$ امکان پذیر است.

۴– $\{1,2,3\}$ رد می شود، زیرا هیچ ترتیب امکان پذیری برای این مجموعه وجود ندارد.

۵– S برابر $\{1,2,4\}$ قرار داده می شود، چون ترتیب $[2,1,4]$ امکان پذیر است.

۶– $\{1,2,4,5\}$ رد می شود، زیرا هیچ ترتیب امکان پذیری برای این مجموعه وجود ندارد.

۷– $\{1,2,4,6\}$ رد می شود، زیرا هیچ ترتیب امکان پذیری برای این مجموعه وجود ندارد.

۸– $\{1,2,4,7\}$ رد می شود، زیرا هیچ ترتیب امکان پذیری برای این مجموعه وجود ندارد.

الگوریتم زمان بندی با مهلت معین،
همواره یک مجموعه بهینه ارائه می کند.

فرادرس

الگوریتم زمان بندی با مهلت معین

```
schedule (n , deadline[ ] , J )
{
  sequence_of_integer   K;
  J =  [1];
  for (i = 2 ;  i <= n ; i++ ) {
        K = J  with  i  added according to nondecreasing values of deadline[i];
        if  ( K  is  feasible)
              J = K;
   }
}
```

آرایه deadline به ترتیب غیر نزولی و بر اساس سود مرتبط با هر کار مرتب سازی شده است.

خروجی تابع، یک ترتیب بهینه(J) برای کارها است.

تحلیل الگوریتم:

عمل اصلی در این الگوریتم، دستور مقایسه است.

کارها در زمان $\theta(n\lg n)$ مرتب شده و به الگوریتم تحویل داده می شوند.

در هر دور تکرار حلقه، حداکثر (i-1) مقایسه برای اضافه کردن کار i ام به k انجام می شود

و حداکثر برای بررسی امکان پذیر بودن k ، به i مقایسه نیاز است.

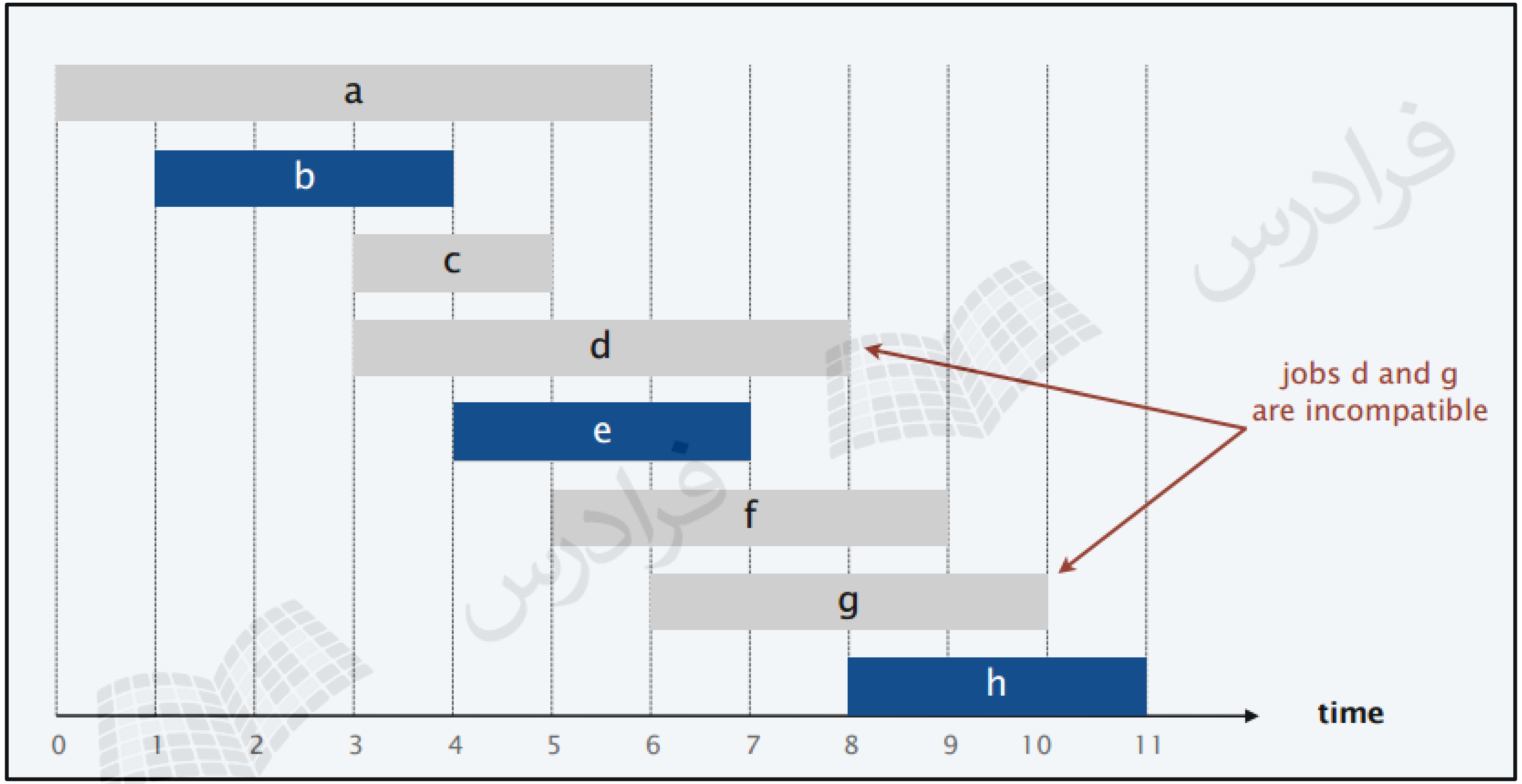
در بدترین حالت داریم:

$$\sum_{i=2}^{n}[(i-1)+i]=n^2-1\in\theta(n^2)$$

مسئله انتخاب فعالیت

activity-selection problem

مسئله انتخاب فعالیت



- Job j starts at s_j and finishes at f_j.
- Two jobs compatible if they don't overlap.
- Goal: find maximum subset of mutually compatible jobs.

مثال

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	8	9	10	11	12	13	14

Some of the compatible activities :

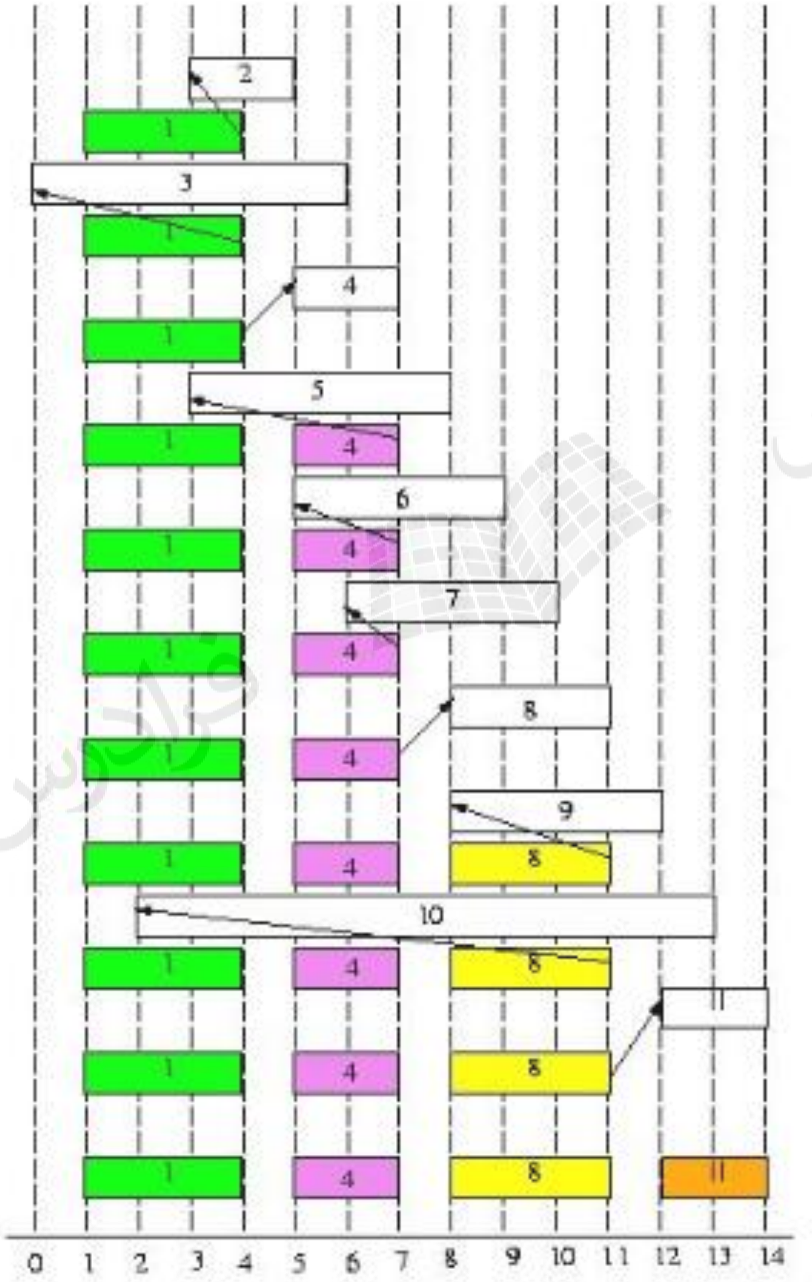
$$\{a_3, a_9, a_{11}\}$$

$$\{a_1, a_4, a_8, a_{11}\}$$

$$\{a_2, a_4, a_9, a_{11}\}$$

فرادرس

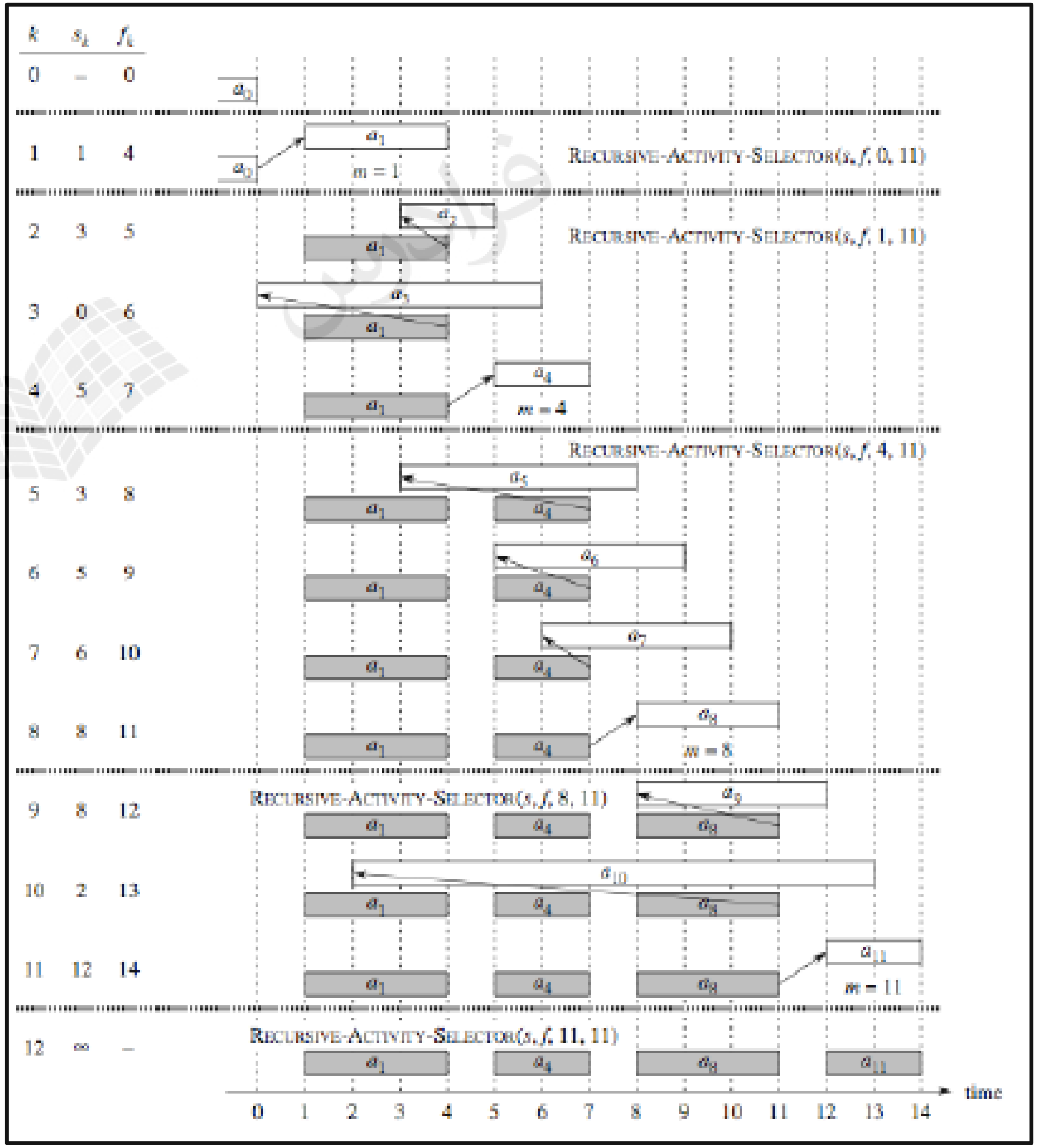
i	s_i	f_i
1	1	4
2	3	5
3	0	6
4	5	7
5	3	8
6	5	9
7	6	10
8	8	11
9	8	12
10	2	13
11	12	14



فرادرس

الگوریتم انتخاب فعالیت

```
RECURSIVE-ACTIVITY-SELECTOR(s, f, i, n)
  m ← i + 1
  while m ≤ n and s_m < f_i        ▷ Find the first activity in S_{i,n+1}.
        do m ← m + 1
  if m ≤ n
     then return {a_m} ∪ RECURSIVE-ACTIVITY-SELECTOR(s, f, m, n)
     else return Ø
```



Greedy Algorithm

قضیه:

Consider any nonempty subproblem S_{ij}, and let a_m be the activity in S_{ij} with the earliest finish time, i.e. $f_m = \min\{f_k : a_k \in S_{ij}\}$. Then

The subproblem S_{im} is empty, so that choosing a_m leaves the subproblem S_{mj} as the only one that may be nonempty.

اثبات:

If $S_{im} \neq \emptyset$ then there exists $a_k \in S_{im}$ such that $f_i \leq s_k < f_k \leq s_m < f_m$. So $a_k \in S_{ij}$ and $f_k < f_m$. ☒

Greedy Algorithm

قضیه:

Consider any nonempty subproblem S_{ij}, and let a_m be the activity in S_{ij} with the earliest finish time, i.e. $f_m = min\{f_k \ : \ a_k \in S_{ij}\}$. Then

Activity a_m is used in some maximum-size subset of compatible activities of S_{ij}.

اثبات:

Suppose that A_{ij} is a maximum-size subset of compatible activities of S_{ij}. Let a_k be the first activity in A_{ij}.

- If $a_k = a_m$, we are done.
- If $a_k \neq a_m$, we construct the subset $A'_{ij} = A_{ij} - \{a_k\} \cup \{a_m\}$. Now, a_m is the first activity in A'_{ij} to finish, and $f_m \leq f_k$. Note that A'_{ij} has the same number of activities as A_{ij}, so A'_{ij} is a maximum-size subset of compatible activities of S_{ij} that includes a_m.

فرادرس

استفاده از روش برنامه نویسی پویا برای مسئله انتخاب فعالیت

Dynamic Programming

C[i , j] : the number of activities in a maximum-size subset of mutually compatible activities in Sij

$$C[i,j] = \begin{cases} 0 & \text{if } i \geq j, \\ \max\limits_{\substack{a_k \in S_{ij} \\ i \leq k \leq j}} \{C[i,k] + C[k,j] + 1\} & \text{if } i < j. \end{cases}$$

$$S \longleftarrow S \cup \{a_0, a_{n+1}\}$$

$$S_{ij} = \{a_k \in S \,:\, f_i \leq s_k < f_k \leq s_j\}$$

$$f_0 = 0$$

$$0 \leq i, j \leq n+1$$

$$s_{n+1} = \infty$$

فرادرس

کد هافمن

Huffman Codes

فرادرس
فرادرس

کد پیشوندی

نوع خاصی از کد با طول متغیر است که کد یک کاراکتر ، آغاز کد کاراکتر دیگر را نمی باشد.

مثلا اگر 01 کد حرف "a" باشد،

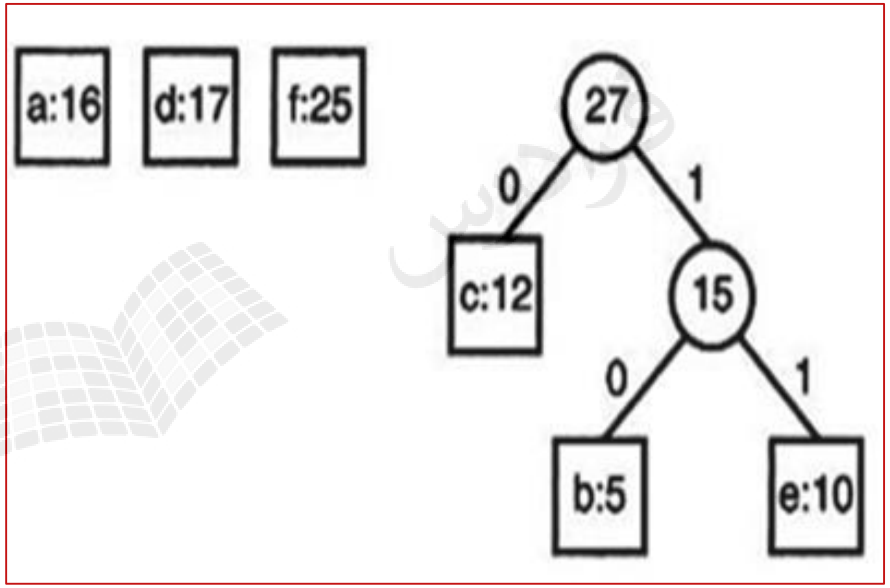
آن گاه 011 نمی تواند کد واژه حرف “b” باشد.

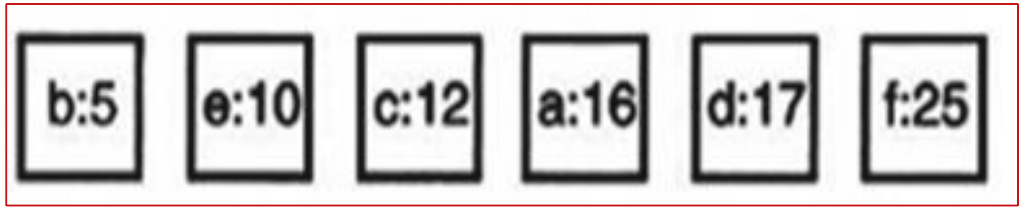
فرادرس
فرادرس

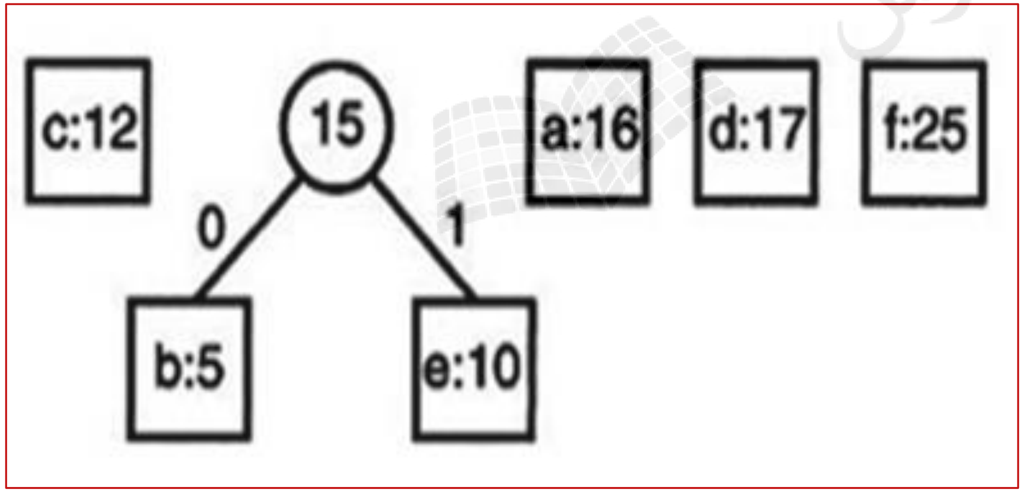
مثال

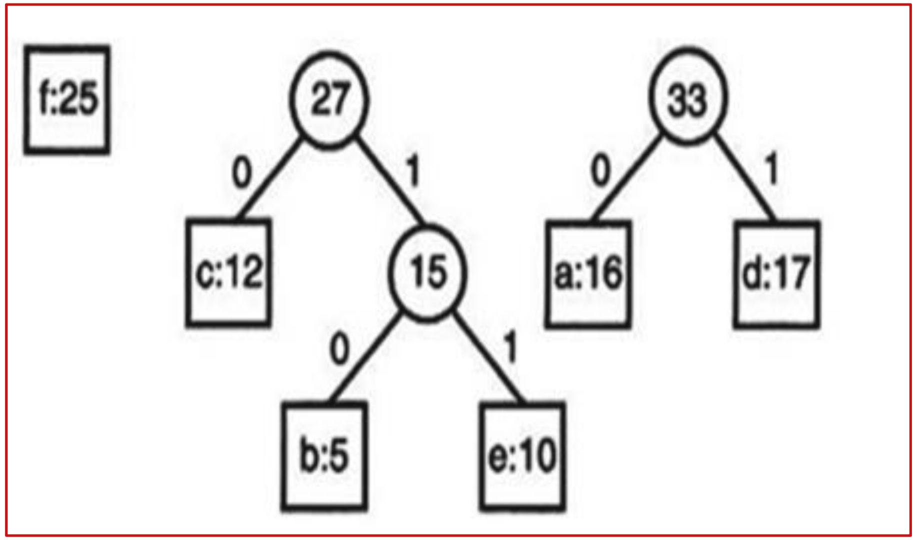
تعداد فراوانی هر کاراکتر در فایل:

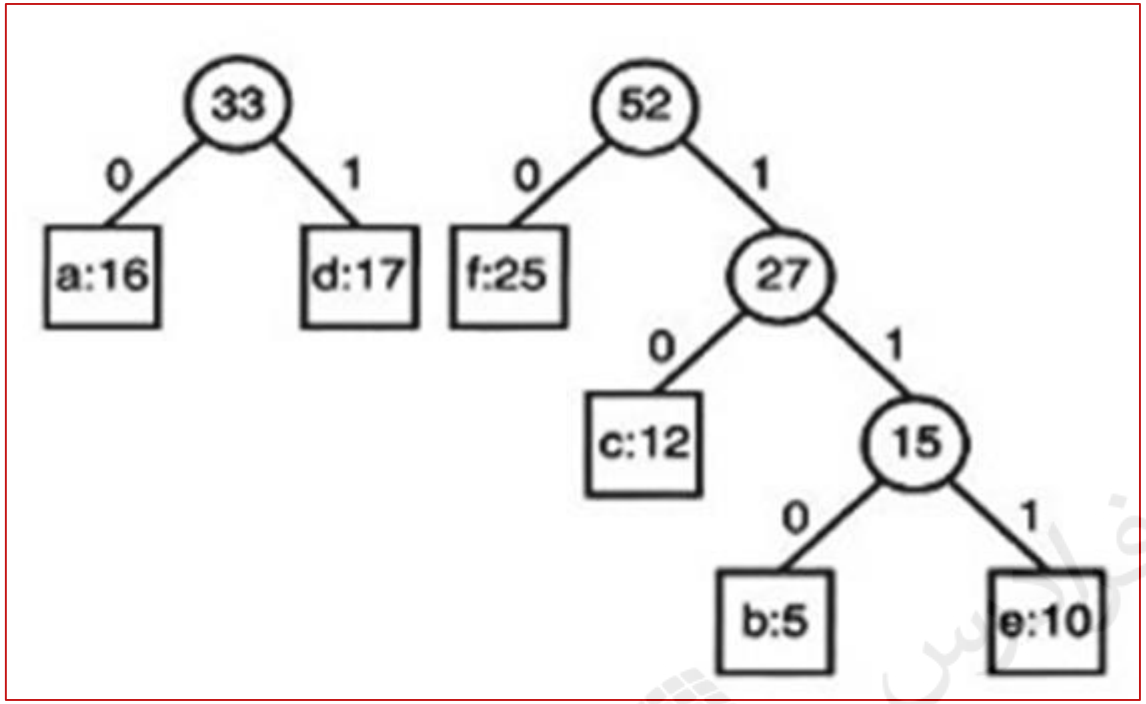
(a:16 , b: 5 , c: 12 , d: 17 , e: 10 , f: 25)

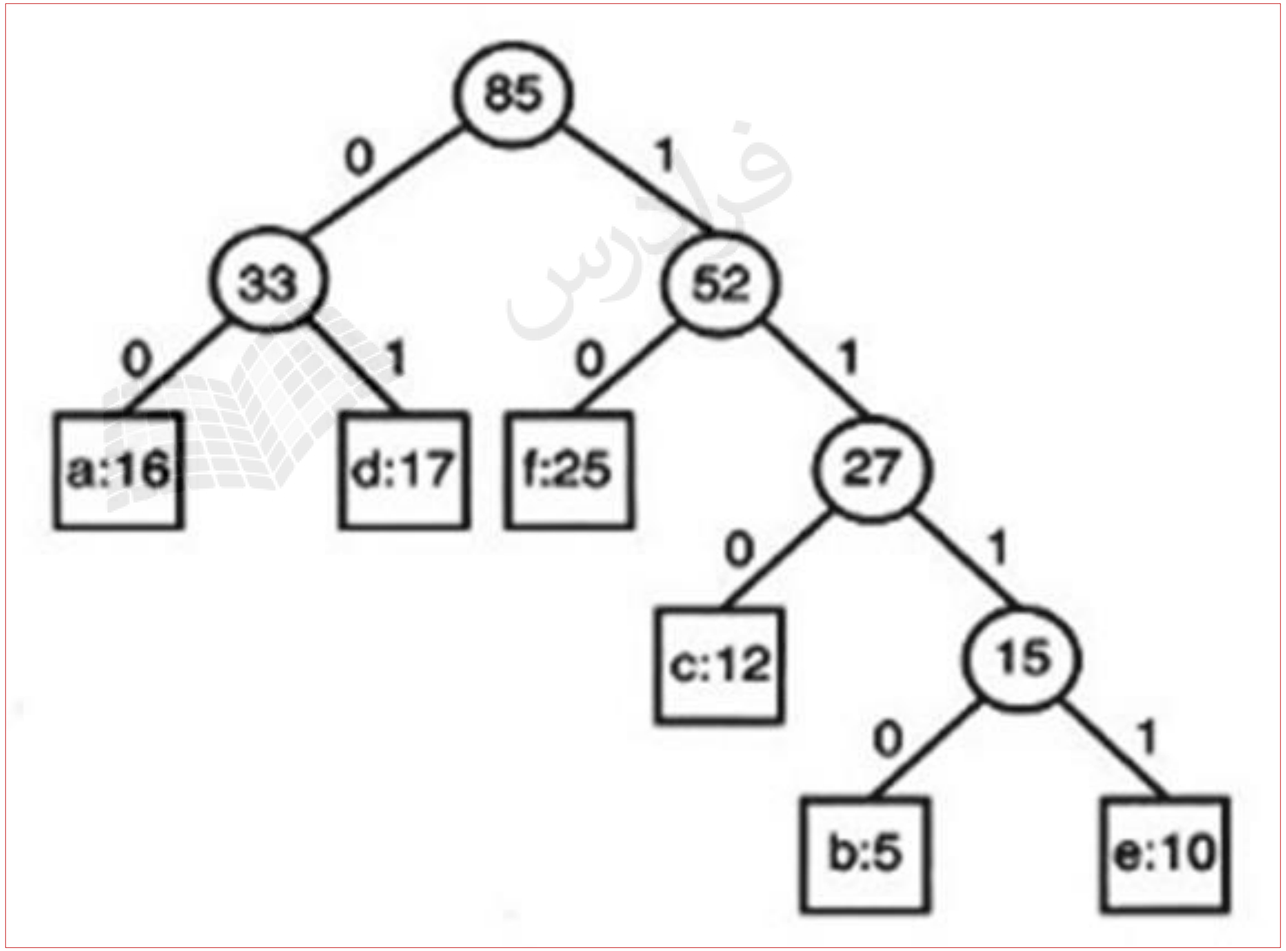












a : 00 **d : 01** **f : 10**

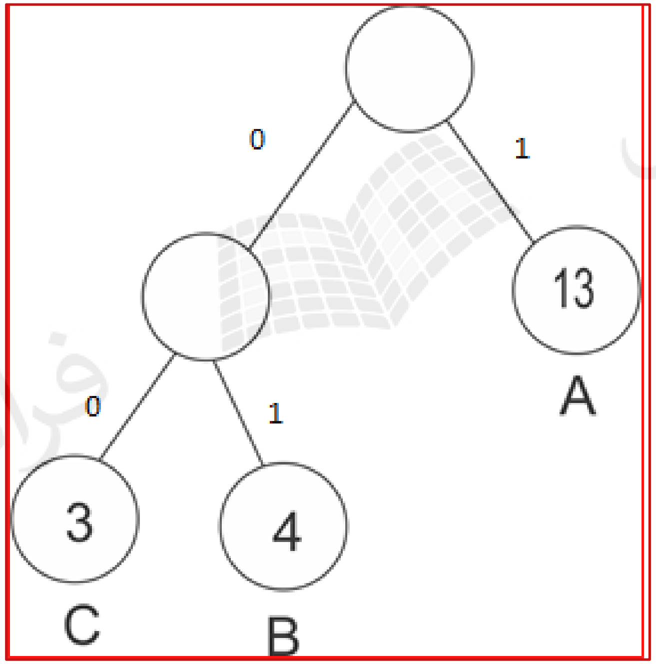
c : 110

b : 1110 **e : 1111**

مثال

کد هافمن عبارت AABAABAACAABAACACABA چند بیت است؟

A: 13 , B: 4 , C: 3



A: 1 , B: 01 , C: 00

بنابراین A یک بیتی و B و C دو بیتی می باشند.

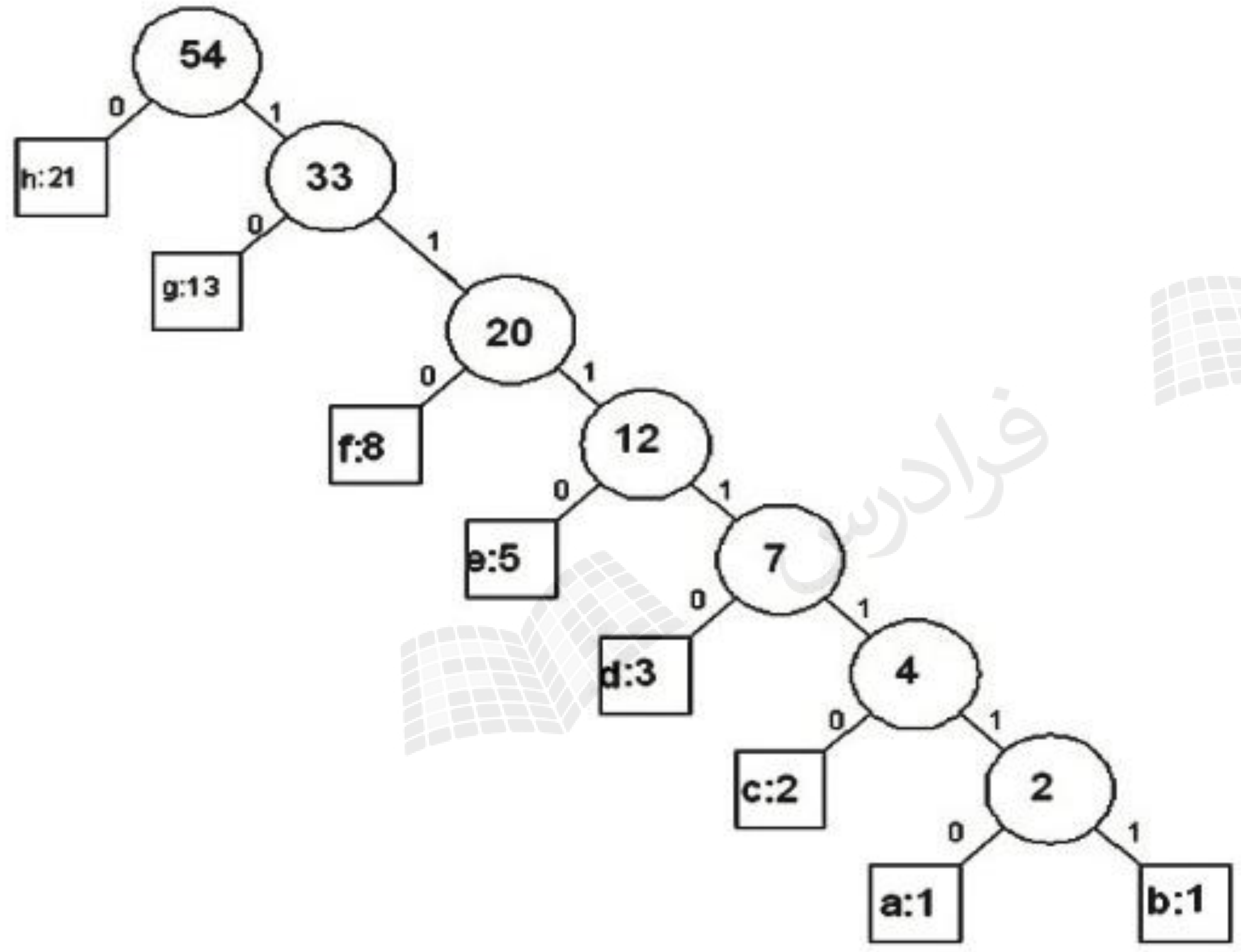
تعداد کل بیت ها در عبارت داده شده بعد از کد گذاری:

$$13\times1+4\times2+3\times2=27$$

فرادرس

مثال

a : 1 , b : 1 , c : 2 , d : 3 , e : 5 , f : 8 , g : 13 , h : 21



کد :

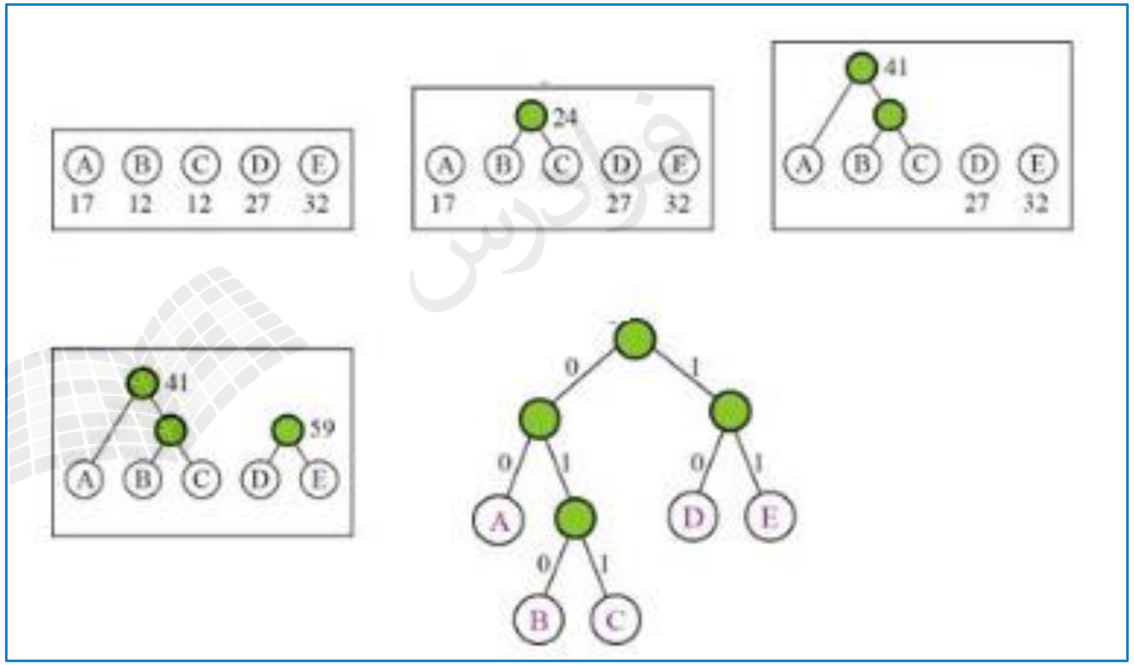
a: 1111110 b : 1111111
c : 111110 d : 11110
e: 1110 f : 110
g : 10 h : 0

به عبارتی:

$a : 1^{n-2}0$, $b : 1^{n-2}1$, $c : 1^{n-3}0$,...

الگوریتم کد هافمن

```
HUFFMAN(C)
n ← |C|
Q ← C
for i ← 1 to n − 1 do
    allocate a new node z
    left[z] ← x ← EXTRACT-MIN(Q)
    right[z] ← y ← EXTRACT-MIN(Q)
    f[z] ← f[x] + f[y]
    INSERT(Q, z)
return EXTRACT-MIN(Q)
```



O(n.logn)

الگوریتم هافمن
یک کد دودویی بهینه
را تولید می کند.

فرادرس

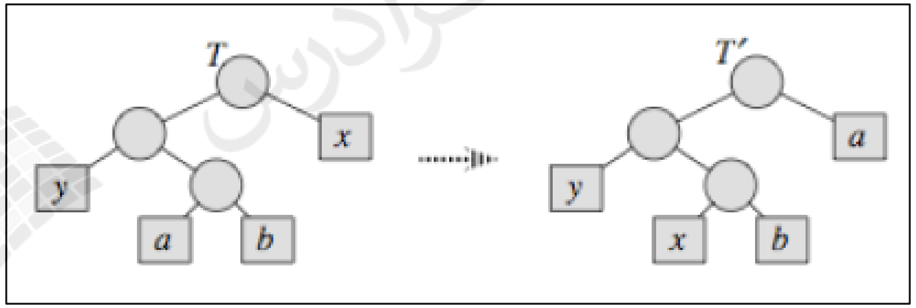
لم

Let x and y be the two least frequent characters.
There is an optimal code tree in which x and y are siblings and have the largest depth of any leaf.

اثبات

Let T be an optimal code tree with depth d.

Since T is a full binary tree, it has at least two leaves a and b at depth d that are siblings ($\{a,b\} \neq \{x,y\}$).



Let T' be the code tree obtained by swapping x and a.

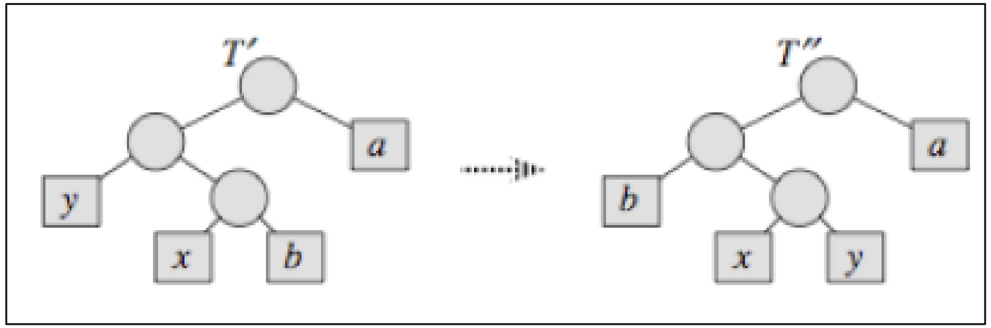
The depth of x (a) increases (decreases) by some amount α, thus $B(T') = B(T) - \alpha[f(a) - f(x)]$.

$$B(T) = \sum_{c \in C} f(c) d_T(c).$$

$f(a) \geq f(x)$ which implies that $B(T') \leq B(T)$.

Since T is optimal, therefore $B(T') = B(T)$ and T' is also optimal.

Swapping y and b yields another optimal code tree T'', where x and y becomes sibling and have the largest depth. □



لم

Let x and y be two characters in C with minimum frequency. Let C' be the alphabet C with characters x, y removed and (new) character z added, so that $C' = C - \{x, y\} \cup \{z\}$.

Define f for C' as for C, except that $f(z) = f(x) + f(y)$.

Let T' be any tree representing an optimal prefix-free code for the alphabet C'.

Then the tree T, obtained from T' by replacing the leaf node for z with an internal node having x and y as children, represents an optimal prefix-free code for the alphabet C.

فرادرس

اثبات

For each $c \in C - \{x, y\}$, we have $d_T(c) = d_{T'}(c)$.

$d_T(x) = d_T(y) = d_{T'}(z) + 1.$

$$f(x)d_T(x) + f(y)d_T(y) = (f(x) + f(y))(d_{T'}(z) + 1) = f(z)d_{T'}(z) + f(x) + f(y)$$

$B(T) = B(T') + f(x) + f(y)$

$B(T') = B(T) - f(x) - f(y).$

suppose that T does not represent an optimal prefix-free code for C

Then there exists a tree T'' such that $B(T'') < B(T)$,

where x and y are siblings in T''.

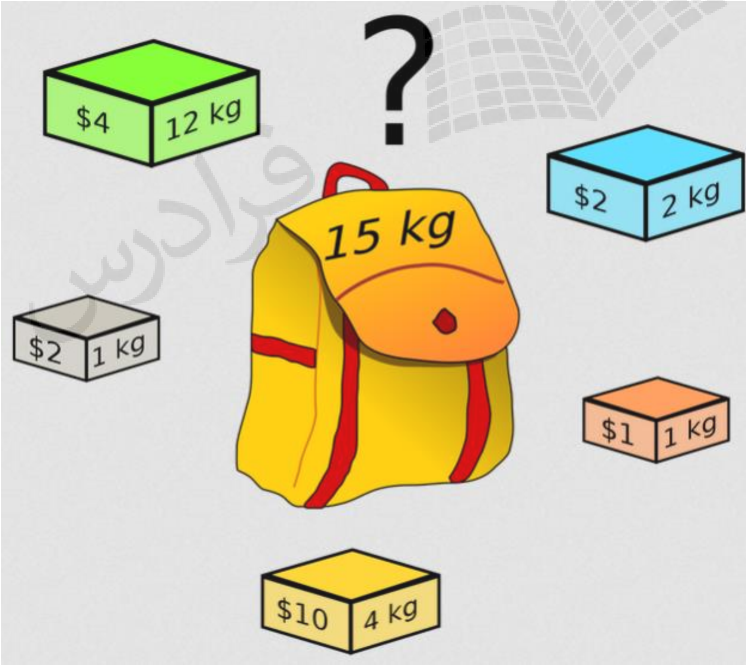
Let T''' be the tree T'' with the common parent of x and y replaced by a leaf z with frequency $f(z) = f(x) + f(y)$.

$$B(T''') = B(T'') - f(x) - f(y) < B(T) - f(x) - f(y) = B(T')$$

This yields a contradiction to the assumption that T' represents an optimal prefix-free code for C'. Thus, T must be an optimal prefix-free code for C.

مسئله کوله پشتی کسری

مسئله کوله پشتی کسری را به کمک یک الگوریتم حریصانه می توان حل کرد.



مثال:

مسئله کوله پشتی کسری (وزن قابل تحمل کوله پشتی : ۳۰ کیلو)

قطعه	ارزش	وزن
A	50	5
B	60	10
C	140	20

راهبرد حریصانه: انتخاب قطعات بر اساس $\frac{p_i}{w_i}$

$$A : \frac{50}{5} = 10 \qquad B : \frac{60}{10} = 6 \qquad C : \frac{140}{20} = 7$$

سود کلی:

$$50 + 140 + \frac{5}{10} \times 60 = 220$$

روش حریصانه برای مسئله کوله پشتی صفر و یک منجر به حل بهینه نمی شود.

قطعه	ارزش	وزن
A	50	5
B	60	10
C	140	20

$$A : \frac{50}{5} = 10 \qquad B : \frac{60}{10} = 6 \qquad C : \frac{140}{20} = 7$$

روش حریصانه : قطعه A و سپس قطعه C انتخاب می شود که سود کل آن 190 تومان است.

روش پویا: اگر قطعه های B و C را انتخاب کنیم، 200 تومان سود حاصل می شود .

فرادرس

مثال

تعیین نحوه انتخاب در مسئله کوله پشتی: (وزن قابل تحمل کوله پشتی : ۳۰ کیلو)

قطعه	ارزش	وزن
A	12	8
B	18	6
C	5	10
D	15	3
E	30	15

$A : \frac{12}{8} = 1.5$ $C : \frac{5}{10} = 0.5$ $E : \frac{30}{15} = 2$

$B : \frac{18}{6} = 3$ $D : \frac{15}{3} = 5$

SORT : D,B,E,A,C

$$15 + 18 + 30 + \frac{6}{8} \times 12 = 72$$

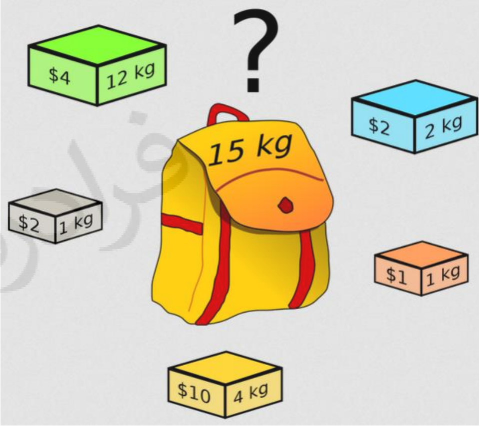
فرادرس

الگوریتم

```
Greedy-fractional-Knapsack-Algorithm(n, b, p[1···n], w[1···n])
{
    X ← 0;                                  Sort objects  p_i/w_i ≥ p_{i+1}/w_{i+1};
    rw ← b;

    for i ← 1 to n do
    {     if w_i < rw   then
          {
              x_i ← 1;
              rw ← rw − w_i;
          }
          else
              break;
    }
    if i < n then x_i ← rw/w_i;
    return(X);
}
```

Sort objects $\frac{p_i}{w_i} \geq \frac{p_{i+1}}{w_{i+1}}$;



مرتبه الگوریتم

```
Greedy-fractional-Knapsack-Algorithm(n, b, p[1···n], w[1···n])
{
    X ← 0;                              Sort objects  p_i/w_i ≥ p_{i+1}/w_{i+1};
    rw ← b;
    for i ← 1 to n do
    {       if w_i < rw   then
            {
                x_i ← 1;
                rw ← rw − w_i;
            }
            else
                break;
    }
    if i < n then x_i ← rw/w_i;
    return(X);
}
```

$$\frac{p_i}{w_i} \geq \frac{p_{i+1}}{w_{i+1}};$$

time complexity is $O(n \log n)$.

قضیه

الگوریتم حریصانه کوله پشتی کسری، همواره یک راه حل بهینه را بر می گرداند.

$$p_1/w_1 \geq p_2/w_2 \geq \cdots \geq p_n/w_n$$

اثبات

$0 \leq x_j < 1$

جواب الگوریتم $X = [1, 1, \cdots, 1, x_j, 0, 0, \cdots, 0]$

جواب بهینه $Y = [y_1, y_2, \cdots y_n]$

فرض کنیم K کوچکترین اندیس است که : $x_k \neq y_k$

ابتدا اثبات می کنیم که : $y_k \leq x_k$

$k < j$: in this case $x_k = 1$ and so $y_k \leq x_k$.

$$Y = [1, \frac{1}{3}, 1, \frac{1}{3}, 0] \qquad k = 2$$

$$X = [1, 1, \frac{2}{3}, 0, 0] \qquad j = 3$$

$k = j$: since $\sum_{i=1}^{n} x_i w_i = b$ so $y_k \leq x_k$

(otherwise $\sum_{i=1}^{n} y_i w_i > b$).

$$Y = [1, 1, \frac{1}{3}, 0, \frac{1}{3}] \qquad k = 3$$

$$X = [1, 1, \frac{2}{3}, 0, 0] \qquad j = 3$$

$k > j$: same as the previous case.

فرادرس

ادامه اثبات

مقدار y_k را زیاد کرده تا برابر x_k شود. جواب به دست آمده را Z می نامم:

$Z = [z_1, z_2, \cdots z_n]$

$$X = [1, 1, \frac{2}{3}, 0, 0]$$

$$Y = [1, \frac{1}{3}, 1, \frac{1}{3}, 0] \qquad k = 2$$

$$Z = [1, 1, , ...]$$

$z_k = x_k$

$$(z_k - y_k)w_k = \sum_{i=k+1}^{n} (y_i - z_i)w_i$$

$$\begin{aligned}
\sum_{i=1}^{n} z_i p_i &= \sum_{i=1}^{n} y_i p_i + (z_k - y_k)p_k - \sum_{i=k+1}^{n} (y_i - z_i)p_i \\
&= \sum_{i=1}^{n} y_i p_i + (z_k - y_k)p_k \frac{w_k}{w_k} - \sum_{i=k+1}^{n} (y_i - z_i)p_i \frac{w_i}{w_i} \\
&\geq \sum_{i=1}^{n} y_i p_i + \frac{p_k}{w_k}\left[(z_k - y_k)w_k - \sum_{i=k+1}^{n} (y_i - z_i)w_i\right]
\end{aligned}$$

$$\sum_{i=1}^{n} z_i p_i \geq \sum_{i=1}^{n} y_i p_i$$

Since Y is an optimal solution, so we have $\sum_{i=1}^{n} z_i p_i = \sum_{i=1}^{n} y_i p_i$.

We can do the same calculation for other indices and finally we obtain $X = Y$.

فرادرس

این اسلاید ها بر مبنای نکات مطرح شده در فرادرس

«آموزش طراحی الگوریتم»

تهیه شده است.

برای کسب اطلاعات بیشتر در مورد این آموزش به لینک زیر مراجعه نمایید

faradars.org/fvsft1092