مرکز آموزش الکترونیکی دانشگاه علم و صنعت ایران

طراحی و تحلیل الگوریتمها

فصل چهارم: روشهای حریصانه

(جلسه اول)

اهداف یادگیری

- آشنایی با کلیات روش حریصانه
- آشنایی با مسائل حل شده با استفاده از روش حریصانه
  - مساله کوله پشتی
  - ادغام دودویی و بهینه فایلها
  - کدگذاری هافمن
  - الگوریتمهای پریم و کراسکال
  - الگوریتم دایکسترا

کلیات

- معمولاً اکر مسائلی که با این روش حل می شوند، دارای n ورودی بوده و هدف یافتن مجموعه از این ورودیهاست که در شرایط معینی صدق کند
- از این روش معمولاً در حل مسائلی استفاده می شود که به دنبال یک مقدار کمینه یا بیشینه هستیم
- بر خلاف روش تقسیم و غلبه که الگوریتم به صورت بازگشتی نوشته می شد، این روش به صورت تکرار شونده مسائل را حل می کند

کلیات (ادامه)

- در این روش مسائل مرحله به مرحله حل می شوند
- در هر مرحله یکی از ورودیها در نظر گرفته شده و بررسی می شود که آیا این ورودی جزء جواب بهینه است یا خیر؟
- در بعضی از مسائل، روش حریصانه صرفاً یک جواب نیمه بهینه ارائه می کند

ساختار عمومی الگوریتمهای حریصانه

```
Algorithm Greedy(A,n)
solution={}
For i=1 to n do
    x=select(A)
    if feasible(x,solution) then
        solution=solution + x
return solution
```

مساله کوله پشتی (knapsack)

- یک کوله پشتی به ظرفیت M و n جسم مفروضند
- وزن جسم i برابر w_i ارزش آن برابر p_i فرض شده است
- اگر کسر x_i ($0<=x_i<=1$) در کوله پشتی قرار گیرد در این صورت سود $p_i x_i$ بدست خواهد آمد
- هدف این اسن که کوله پشتی را با اجسامی پر کنیم که سود کلی حاصل از آنها مازیمم شود

مساله کوله پشتی(ادامه)

* در واقع هدف ماکزیمم کردن کمیت زیر است:

$$\sum_{i=1}^{n} p_i x_i$$

* به شرطی که مجموع وزنهای اجسام انتخاب شده از ظرفیت کوله پشتی بیشتر نشود:

$$\sum_{i=1}^{n} w_i x_i \leq M$$

مثال

❊ نمونه مساله کوله پشتی زیر را در نظر بگیرید:

$(p_1,p_2,p_3)=(25,24,15)$, $(w_1,w_2,w_3)=(18,15,10)$,
$M=20$, $n=3$

❊ چند جواب امکان پذیر (و نه بهینه) به صورت زیر است:

(1/2, 1/3, 1/4)	حجم=16.5	سود=24.25
(0, 2/3, 1)	حجم=20	سود=31
(0, 1, 1/2)	حجم=20	سود=31.5

راه حل حریصانه

- ابتدا جسمی را انتخاب کنیم که نسبت سود به وزنش بیشینه باشد
- و آنقدر ادامه دهیم تا کوله پشتی پر شود
- ممکن است در آخرین انتخاب نتوان همه جسم مورد نظر را در کوله پشتی جای داد، در این حالت کسری از جسم انتخاب می شود
- برای این منظور کافی است اجسام را بر اساس نسبت سود به وزنشان نزولی مرتب کنیم

مثال

$(p_1/w1, p_2/w2, p_3/w3)=(25/18,24/15,15/10)$,

$\rightarrow$ $(p_2/w2 , p_3/w3, p_1/w1)$

* انتخاب جسم دوم: $w_1=15<20$،حجم باقیمانده 20-15=5
* چون وزن سومین جسم برابر ۱۰ و بزرگتر از ۵ است پس به مقدار 5/10 از آنرا انتخاب می کنیم
* سود حاصل: 1*24+5/10*15=31.5

الگوریتم حریصانه برای مساله کوله پشتی

```
Algorithm Greedy-knapsack(p,w,M,x,n,profit)
Profit=0; cu=M
For i=1 to n do{
  if w[i]>cu then exit
   x[i]=1;
   cu=cu-w[i];
   propit=profit+p[i];
}
X[i]=cu/w[i];
Profit=profit+p[i]*x[i]
```

مثال

* مساله کوله پشتی زیر را حل کنید:

$P=(p_1,p_2,p_3,p_4,p_5,p_6)=(10,7,12,13,6,20)$

$W=(w_1,w_2,w_3,w_4,w_5,w_6)=(2,1,3,2,12,8)$

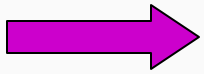
$M=14$

$N=6$

$P/W=(5,7,4,6.5, 0.5, 3.5)$

مثال (ادامه)

i	w_i	p_i	Pi/Wi	x_i	Cu	profit
2	1	7	7	1	13	7
4	2	13	6.5	1	11	20
1	2	10	5	1	9	30
3	3	12	4	1	6	42
6	8	20	2.5	0.75	0	57
5	12	6	0.5	0	0	57



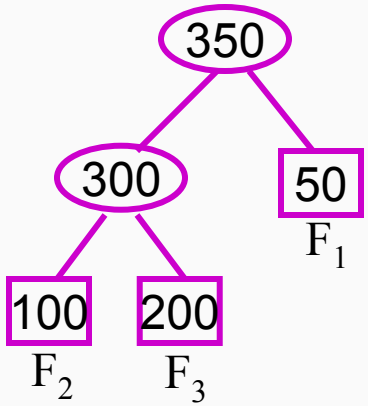
x=(1,1,1,1,0,0.75)

ادغام دودویی و بهینه فایلها

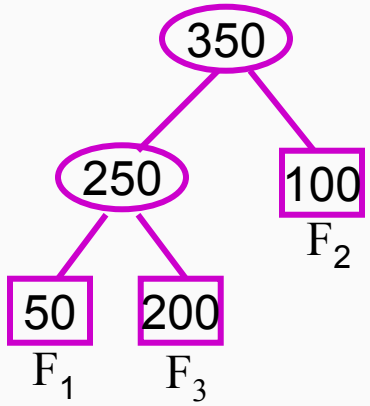
- تعداد $n>2$ فایل را که هر یک حاوی تعدادی عدد هستند در نظر بگیرید. فرض کنید هر یک از فایلها به صورت صعودی مرتب شده اند. حال می خواهیم آنها را دوتا دوتا با هم ادغام کنیم تا فایل مرتب شده واحدی بدست آید
- اگر فایل A دارای k_1 عنصر و فایل B دارای k_2 عنصر باشند، آنگاه برای ادغام این دو فایل نیاز به k_1+k_2 مقایسه می باشد
- هدف یافتن شیوه ای برای ادغام است که در کل کمترین تعداد مقایسه صورت گرفته باشد

مثال

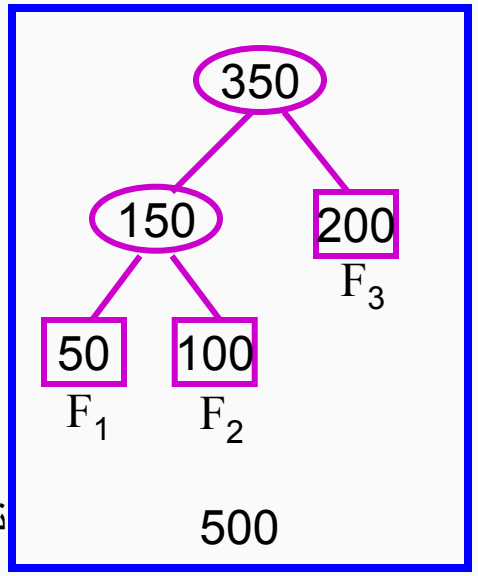
* سه فایل مرتب شده F_1، F_2و F_3 به ترتیب با اندازه های ۵۰، ۱۰۰ و ۲۰۰ را در نظر بگیرید.



تعداد کل مقایسات= 650=350+300



تعداد کل مقایسات= 600=350+250



Weighted External Path Length=WEPL=$\sum_{i=1}^{n} d_i q_i$

الگوریتم حریصانه برای ادغام دودویی بهینه

- ابتدا فایلها را بر اساس اندازه شان صعودی مرتب کنید
- سپس تا زمانیکه تمام فایلها با هم ادغام نشده اند، دو فایل حاوی کمترین عناصر را با هم ادغام کنید و فایل جدید را در میان سایر فایلها درج نمایید

الگوریتم حریصانه برای ادغام دودویی و بهینه

```
Algorithm Tree(L,n)
For i=1 to n-1 do
{ GetNode(T)
   Lchild(T)=Least(L)
   Rchild(T)=Least(L)
   Weight(T)=Weight(Lchild(T)+Weight(Rchild(T))
   Insert(T,L)
}
```

O(1)

O(1)

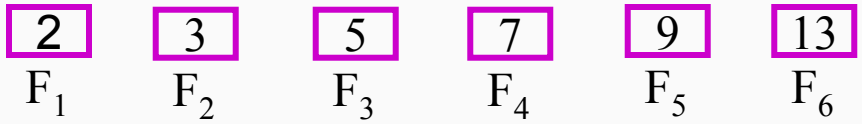
O(log n)

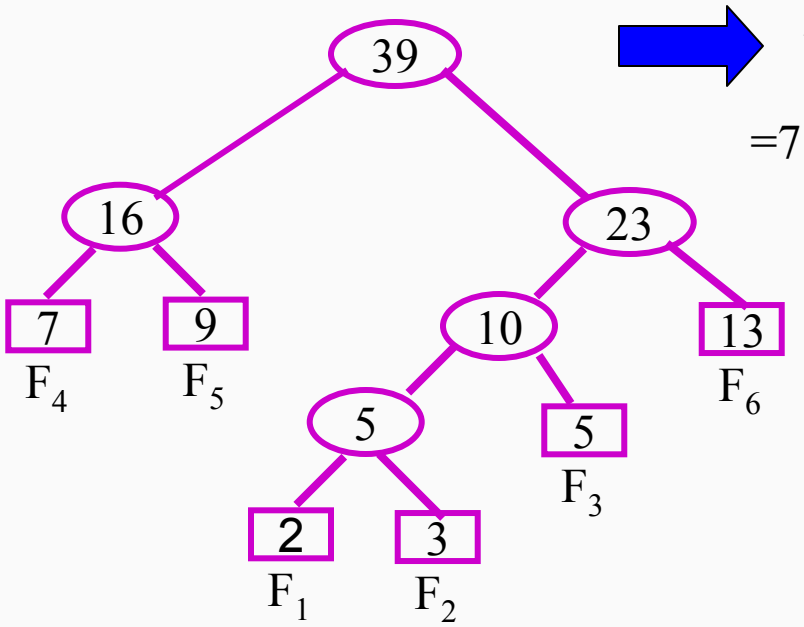
O(1)

O(n log n)

مثال

* ساختار L در شروع:





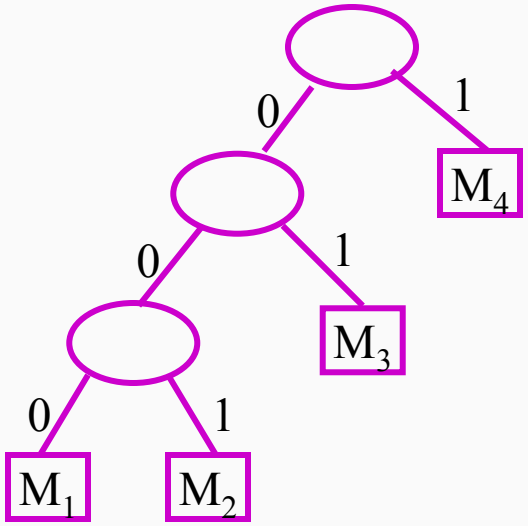
WEPL=16+5+10+23+39=93

=7*2+9*2+2*4+3*4+5*3+13*2

کدهای هافمن

- کاربرد دیگر درخت دودویی با WEPL مینیمم، بدست آوردن یک مجموعه بهینه از کدها برای n پیغام $M_1,M_2,\ldots,M_n$ می باشد
- کدهای بدست آمده با این روش صرفاً متشکل از ۰ و ۱ می باشند
- هدف یافتن کدهایی است که طولشان کمینه باشد

روش کدگذاری هافمن



M_1=000 M_2=001 M_3=01 M_4=1

متن=$M_2M_2M_1M_2M_1M_2M_3M_4M_3M_1$

متن رمز شده= 001001000001000000101101000

طول رمز= 26

مثال

* متنی شامل ۷۰۰۰ حرف از حروف a,b,c,d,e,f مفروض است

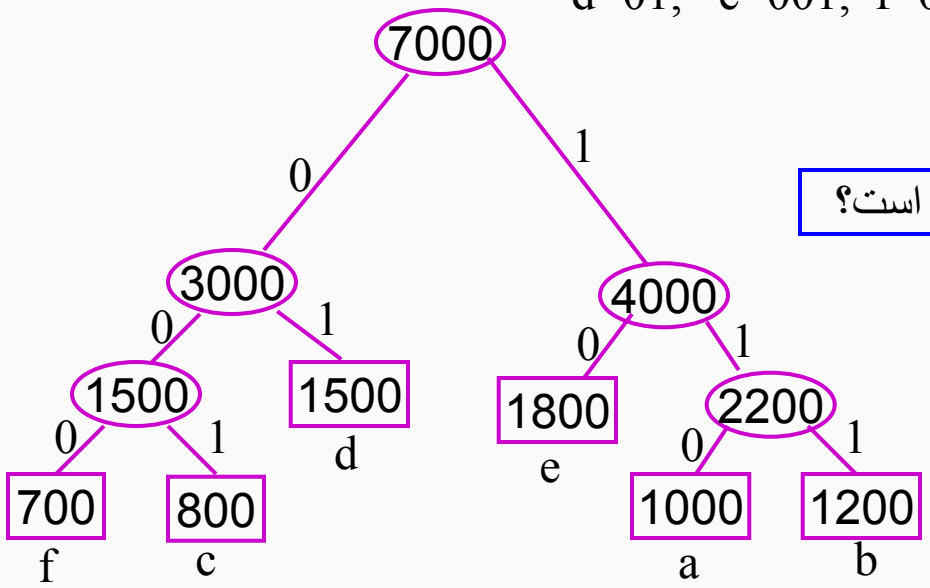
a=110, b=111, c=001,

d=01, e=001, f=000

a=1000, b=1200,

c=800 , d=1500,

e=1800, f=700



کمترین تعداد بیت لازم برای کد کردن فایل فوق چقدر است؟

WEPL=700*3+800*3+1500*2+1800*2+
1000*3+1200*3=17700

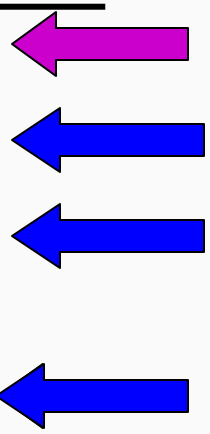
مساله زمانبندی فعالیتها

- فرض کنید که مجموعه ای از n فعالیت پیشنهاد شده است که می خواهند از یک منبع واحد استفاده کنند
- در یک زمان خاص فقط یک فعالیت می تواند از این منبع استفاده کند
- هر فعالیت i دارای یک زمان شروع S_i و یک زمان خاتمه f_i می باشد

مثال

* هدف انتخاب بیشترین تعداد فعالیت قابل انجام است

فعالیت	زمان شروع	زمان خاتمه
1	3	5
2	1	4
3	4	6
4	3	7
5	6	9



الگوریتم حریصانه برای مساله انتخاب فعالیتها

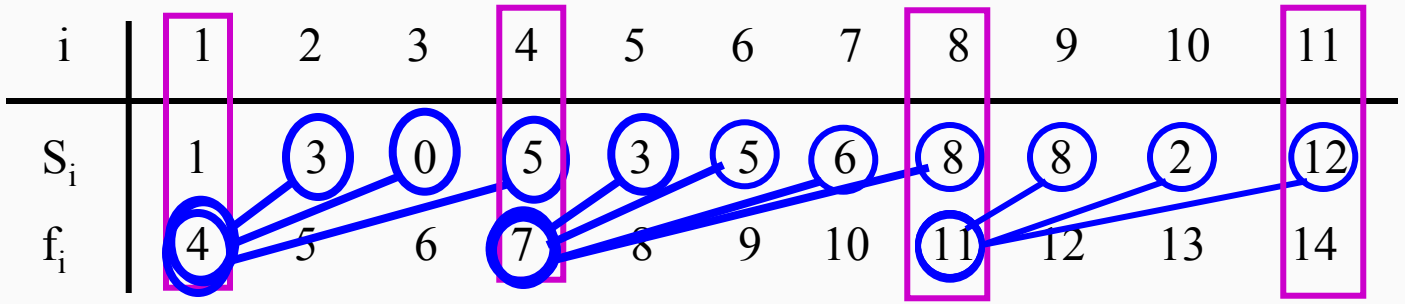
```
Algorithm activity_Selection(s,f)
A={1}; j=1;
For i=2 to n do
    if s[i]>=f[j] then
        A=A+{i}
        j=i;
Return A;
```

f به صورت صعودی مرتب شده است

زمان مرتب سازی

$\Theta(n) \rightarrow \Theta(n+n\log n)= \Theta(n\log n)$

مثال



پاسخ: مجموعه فعالیتهای 1، 4، 8 و 11

زمانبندی فعالیتها با مهلت

- برای فعالیت i یک مهلت صحیح $d_i>=0$ و یک سود $p_i>0$ در نظر گرفته می شود و مدت زمان انجام آن برابر واحد در نظر گرفته میشود
- برای فعالیت i زمانی سود p_i حاصل می شود که در مهلت d_i انجام شود
- ضمناً در هر لحظه فقط یک فعالیت می تواند انجام شود
- هدف یافتن مجموعه ای از فعالیتهاست که بتوان آنها را در مهلت تعیین شده انجام داد و مجموع سود حاصل از آنها نیز بیشینه باشد

مثال

$(p_1,p_2,p_3,p_4)=(100,10,15,27)n=4$ $(d_1,d_2,d_3,d_4)=(2,1,2,1)$

مقدار	ترتیب پردازش	جواب امکان پذیر
110	2و1	(1,2)
115	1و3 یا 3و1	(1,3)
127	4و1	(1,4)
25	2و3	(2,3)

الگوریتم حریصانه برای زمانبندی با مهلت

Algorithm Greedyjob(d,j,n)

j={1}

for i=2 to n do

if همه فعالیتها در j به همراه i در مهلت خود قابل انجام اند then

j=j+{i}

$O(n^2)$

مثال

$(p_1,p_2,p_3,p_4,p_5)=(20,15,10,5,1)$,

$(d_1,d_2,d_3,d_4,d_5)=(2,2,1,3,3)$, $n=5$

j	بازه زمانی	فعالیت مورد بررسی	ترتیب قابل اجرا	ارزش
{}	هیچ	1	1	20
{1}	[1,2]	2	1و2 یا 2و1	35
{1,2}	هیچ	3	هیچ	35
{1,2}	هیچ	4	1و2و3	40
{1,2,3}	هیچ	5	هیچ	40