طراحی الگوریتم ها

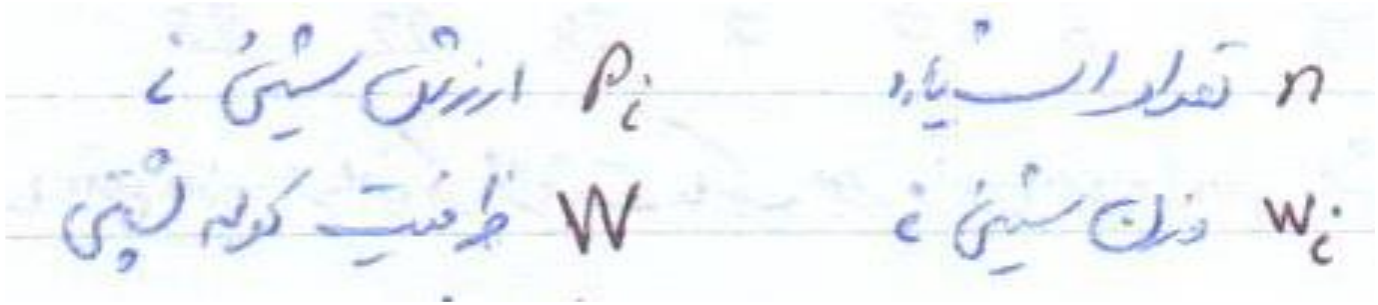
روش برنامه ریزی پویا(ادامه)

استاد درس: مهدی جبل عاملی

کوله پشتی (Knapsack)

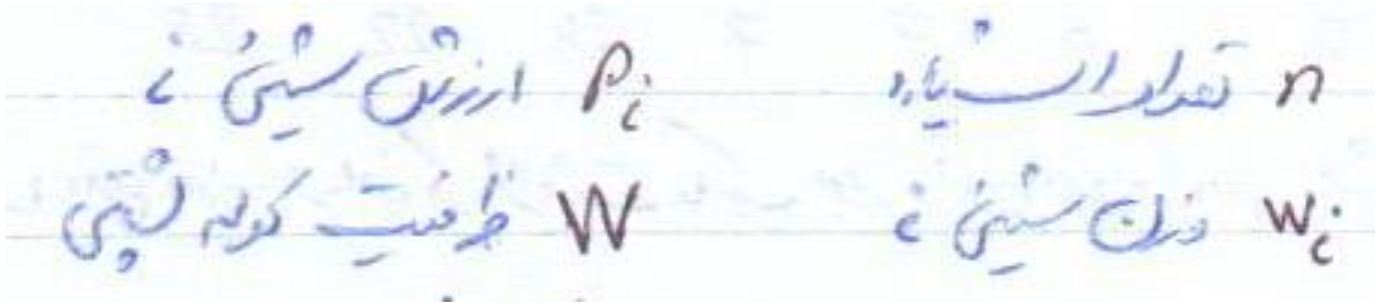
کوله پشتی (Knapsack)

- ورودی ها:

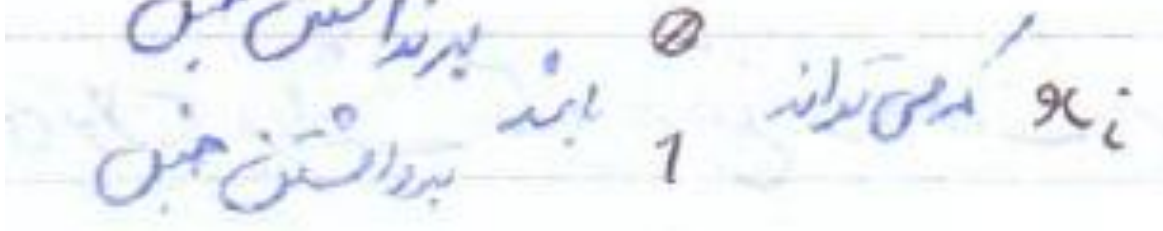


کوله پشتی (Knapsack)

- ورودی ها:

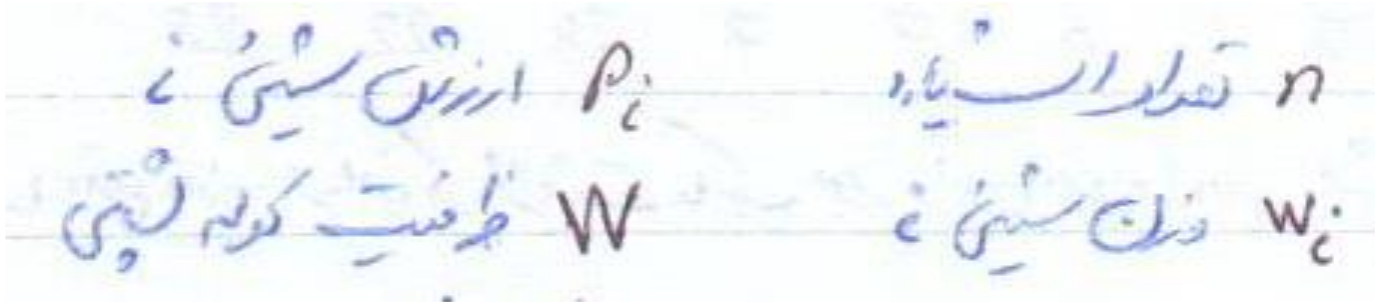


- خروجی:

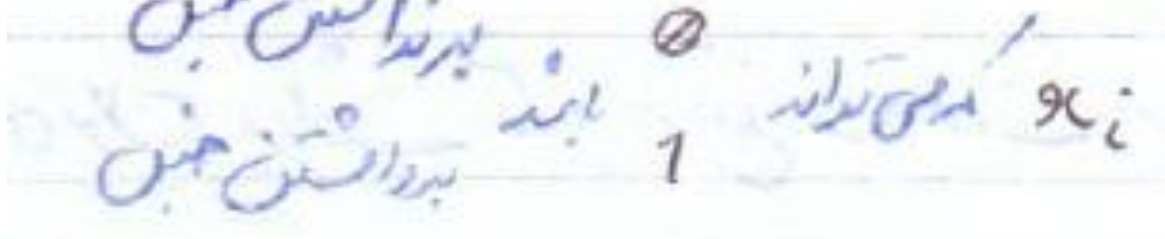


کوله پشتی (Knapsack)

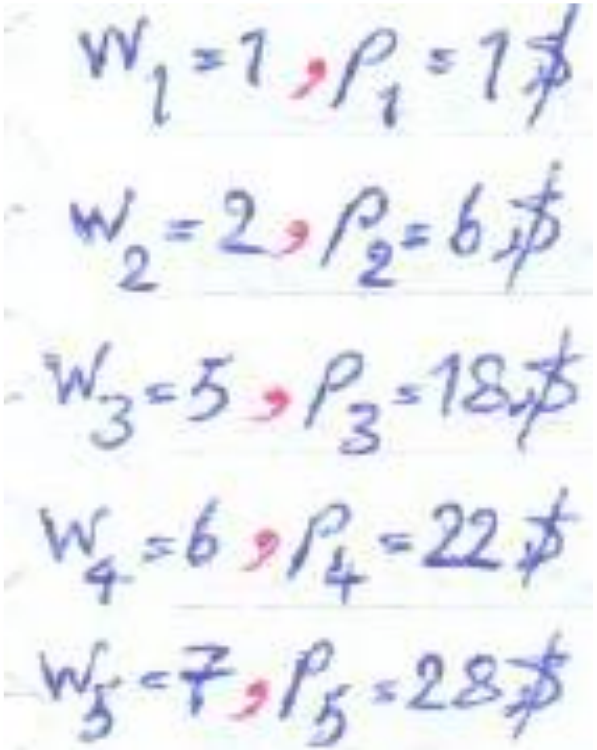
- ورودی ها:

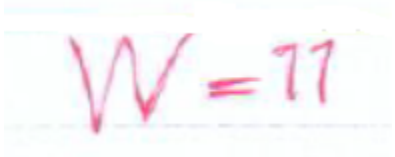


- خروجی:

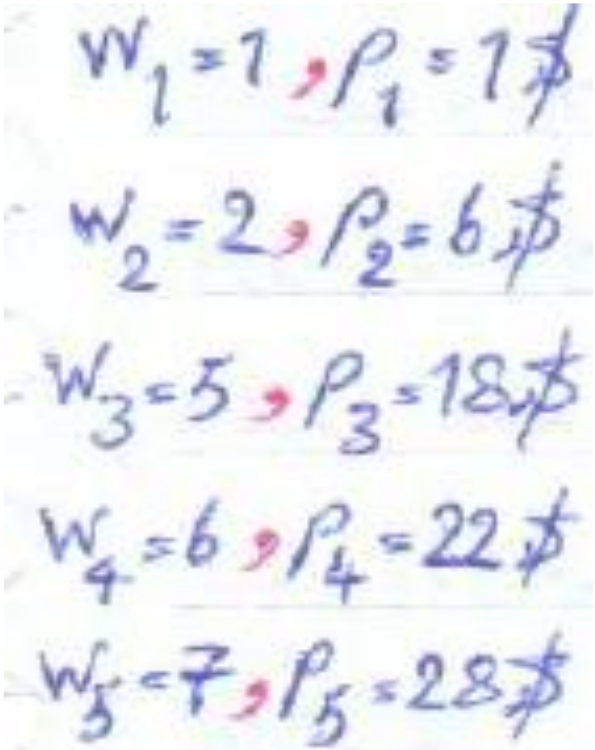


- نمونه مساله:





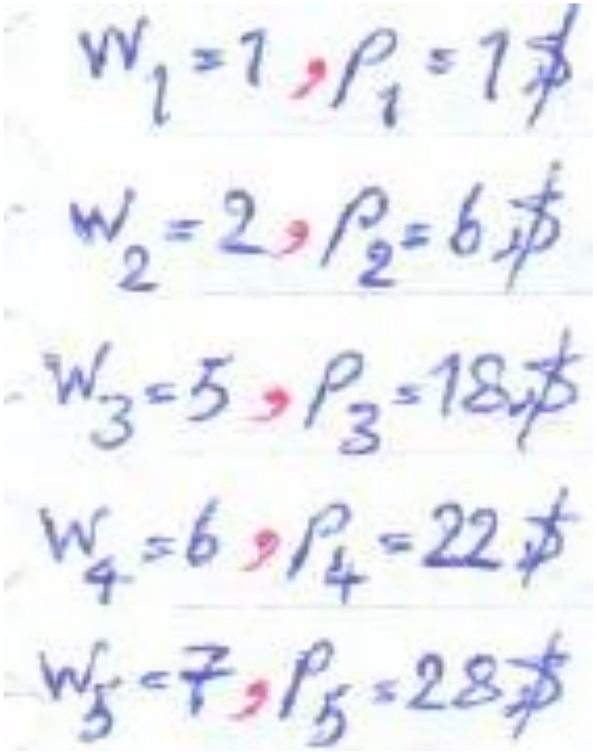
- نمونه مساله:

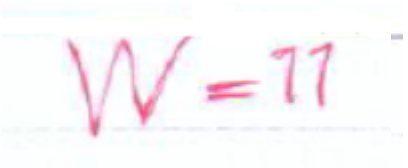


$W = 11$

$$\binom{n}{0} + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n} = 2^n$$

- ایده حل:


$w_1 = 1 \Rightarrow p_1 = 1$
$w_2 = 2 \Rightarrow p_2 = 6$
$w_3 = 5 \Rightarrow p_3 = 18$
$w_4 = 6 \Rightarrow p_4 = 22$
$w_5 = 7 \Rightarrow p_5 = 28$


$W = 77$

	0	1	2	3	4	5	6	7	...	11
$w_1=1, p_1=1\$$	0$	1$	1$	1$	1$	1$	1$	1$		1$
$w_2=2, p_2=6\$$	0$	1	6	6+1	6+1	6+1	6+1	6+1		6+1
$w_3=5, p_3=18\$$	0$	1	6	7	7	18	18+1	18+6		18+7
$w_4=6, p_4=22\$$	0$	1	6	7	7	18	22	~~22+1~~ 24		22+18
$w_5=7, p_5=28\$$	0$	1	6	7	7	18	22	28		~~28+7~~ 40

	0	1	2	3	4	5	6	7	...	11
$w_1=1, P_1=1$$	0$	1$	1$	1$	1$	1$	1$	1$		1$
$w_2=2, P_2=6$$	0$	1	6	6+1	6+1	6+1	6+1	6+1		6+1
$w_3=5, P_3=18$$	0$	1	6	7	7	18	18+1	18+6		18+7
$w_4=6, P_4=22$$	0$	1	6	7	7	18	22	~~22+1~~ 24		22+18
$w_5=7, P_5=28$$	0$	1	6	7	7	18	22	28		~~28+7~~ 40

$$B[i,j]=\begin{cases} 0 & i=1,\ w_i>j \\ P_1 & i=1,\ \text{else} \\ B[i-1,j] & w_i>j,\ \text{else} \\ \max\{P_i+B[i-1,j-w_i],\ B[i-1,j]\} & \text{else} \end{cases}$$

```
Knapsack(n, w[1..n], p[1..n], W){
        for i=1 to n do
                for j= 0 to W do
```

$$B[i,j] = \begin{cases} 0 & i=1,\ w_i > j \\ p_1 & i=1,\ \text{else} \\ B[i-1,j] & w_i > j,\ \text{else} \\ \max\{p_i + B[i-1, j-w_i],\ B[i-1,j]\} & \text{else} \end{cases}$$

```
return  B[n,W]
}
```

```
Knapsack(n, w[1..n], p[1..n], W){
        for i=1 to n do
                for j= 0 to W do
```

$$B[i,j] = \begin{cases} 0 & i=1 \text{ , } w_i > j \\ P_1 & i=1 \text{ , else} \\ B[i-1,j] & w_i > j \text{ , else} \\ \max\{P_i + B[i-1, j-w_i], B[i-1,j]\} & \text{else} \end{cases}$$

```
return  B[n,W]
}
```

زمان اجرا: $\in \theta(nW)$ مصرف حافظه فعلی: $n \times (W+1)$

- بهبود مصرف حافظه:
  - چون مقادیر هر سطر فقط به سطر قبل وابسته است می توان مصرف حافظه را به یک سطر کاهش داد به شرطی که مقادیر هر سطر را از آخر به اول محاسبه کنید و در همان سطر ذخیره کنید در این صورت مصرف حافظه به اندازه W+1 خواهد بود.

Making change — خرد کردن پول

خرد کردن پول یا Making change

- ورودی ها:

n تنوع سکه هایی که داریم c_i مبلغ سکه نوع i

A مبلغ پول کل

خرد کردن پول با Making change

- ورودی ها:

n تنوع سکه هایی که داریم
c_i مبلغ سکه نوع i
A مبلغ پول کل

- خروجی:

x_i ها تعداد سکه‌های ما هستند و می‌تواند 0، 1، 2، ... باشد

خرد کردن پول با Making change

- ورودی ها:

n تنوع سکه هایی که داریم
c_i مبلغ سکه نوع i
A مبلغ پول کل

- خروجی:

x_i ها تعداد سکه‌های ما هستند و می‌تواند 0، 1، 2، ... باشد

1. $\sum_{i=1}^{n} x_i \cdot c_i = A$

2. Minimize $\sum_{i=1}^{n} x_i$

- ایده حل:

$C_1 = 1\$$

$C_2 = 3\$$

$C_3 = 4\$$

$A = 6\$$

B	مبلغ						
	0 $	1 $	2 $	3 $	4 $	5 $	6 $
$C_1 = 1\$$	∅	1	1+1	1+2	1+3	1+4	1+5
$C_2 = 3\$$	∅	1	2	1	1+1	1+2	1+1
$C_3 = 4\$$	∅	1	2	1	1	1+1	1+2
$A = 6\$$							

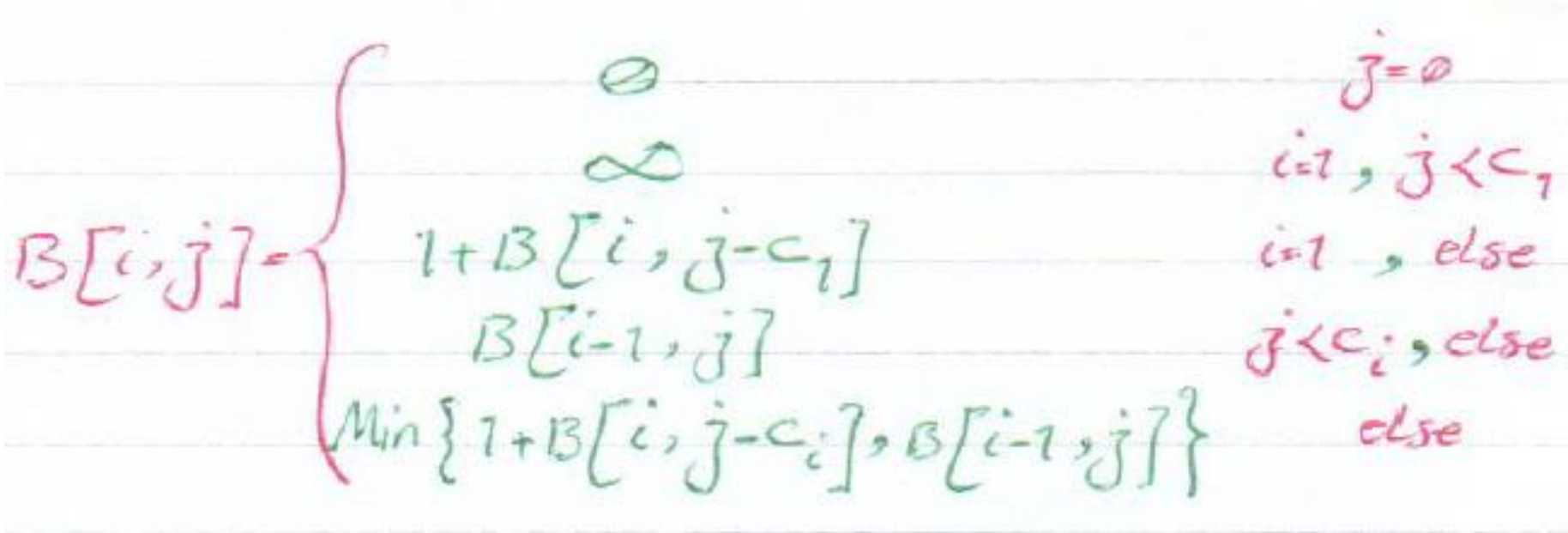
15

2

B	مبلغ 0$	1$	2$	3$	4$	5$	6$
$C_1 = 1\$$	∅	1	1+1	1+2	1+3	1+4	1+5
$C_2 = 3\$$	∅	1	2	1	1+1	1+2	1+1
$C_3 = 4\$$	∅	1	2	1	1	1+1	1+2
$A = 6\$$							

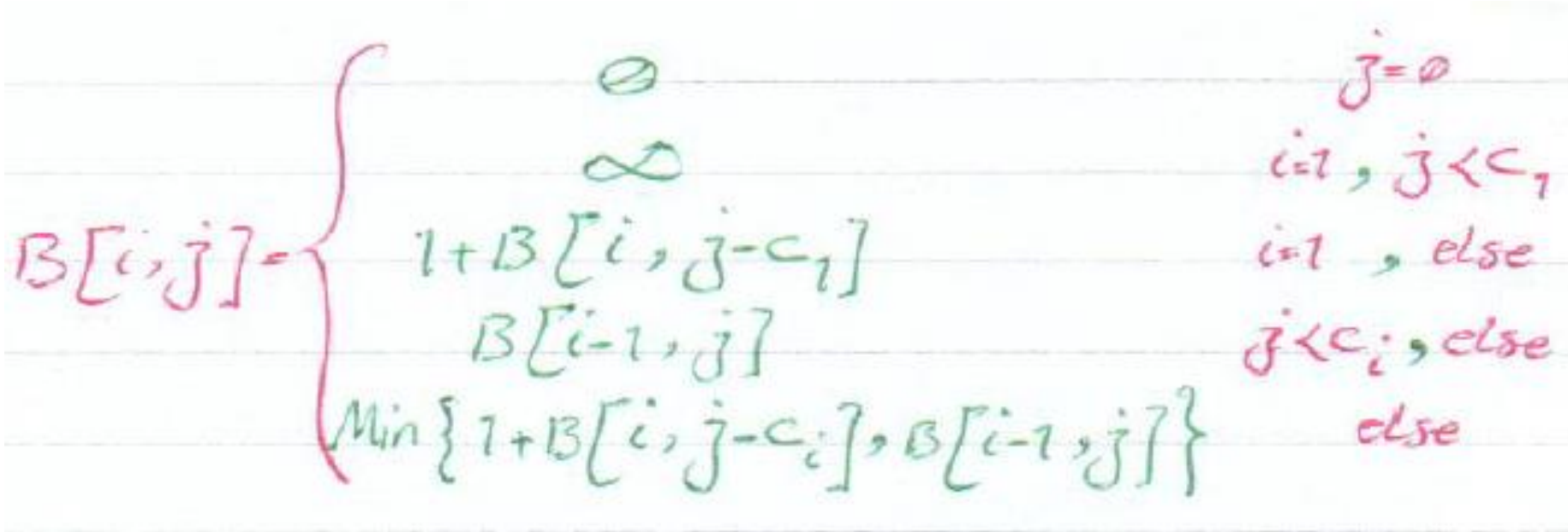
$$B[i,j] = \begin{cases} \emptyset & j = 0 \\ \infty & i = 1,\ j < C_1 \\ 1 + B[i, j - C_1] & i = 1,\ \text{else} \\ B[i-1, j] & j < C_i,\ \text{else} \\ \text{Min}\{1 + B[i, j - C_i], B[i-1, j]\} & \text{else} \end{cases}$$

```
Making_change(n, c[1..n], A){
        for i=1 to n do
                for j= 0 to A do
```




```
return  B[n,A]
}
```

```
Making_change(n, c[1..n], A){
        for i=1 to n do
                for j= 0 to A do
```




```
return  B[n,A]
}
```

زمان اجرا: $\theta(nA)$ مصرف حافظه فعلی: $n(A+1)$

- بهبود مصرف حافظه:
- چون مقادیر هر سطر فقط به سطر قبل وابسته است می توان مصرف حافظه را به یک سطر کاهش داد در این صورت مصرف حافظه به اندازه A+1 خواهد بود.

تمرین:

- در الگوریتم کوله پشتی، ترتیب اشیاء داده شده را به هم بریزید و الگوریتم را دوباره اجرا کنید.

- الگوریتم خرد کردن پول را برای مبلغ A=9 و سکه های {1, 4, 6} اجرا کنید.