مساله کوله پشتی ۰-۱

- تفاوت با مسایل قبلی: بهینه سازی
  - تا زمانی که جستجو به پایان نرسد نمی توان دریافت که آیا گره ای حاوی یک راه حل می باشد یا خیر.
  - در حل کردن مسایل بهینه سازی توسط عقبگرد، همواره فرزندان یک گره امید بخش را ملاقات می کنیم.

الگوریتم کلی

```
void checknode ( node v )
{
   node u ;

   if ( value (v)  is better than best )
         best = value (v) ;
   if ( promising (v))
         for ( each child u of v )
               checknode (u) ;
}
```

کوله پشتی

- گره غیر امید بخش

$$weight \geq W$$

- مرتب سازی قطعات برحسب p_i / w_i به صورت غیر نزولی
- تعیین حد بالا برای سود قابل حصول از گسترش دادن گره
  - $profit$ حاصل جمع ارزش قطعاتی که تا آن گره در نظر گرفته شده اند
  - $weight$ حاصل جمع اوزان آن قطعات
  - مقدار اولیه متغیر $bound$ را برابر $profit$ قرار می دهیم
  - مقدار اولیه متغیر $totweight$ را برابر $weight$ قرار می دهیم
  - هر بار به روش حریصانه یک قطعه برداشته و ارزش آن را به $bound$ و وزنش را به $totweight$ اضافه می کنیم تا به قطعه ای برسیم که در صورت برداشتن آن $totweight$ از W بیشتر شود. در این صورت کسری از آن را با توجه به ظرفیت باقیمانده برداشته و ارزش آن کسر را به bound اضافه می کنیم.

کوله پشتی

- فرض کنید گره در سطح i باشد و گره واقع در سطح k، گره ای باشد که حاصل جمع اوزان را از W بیشتر کند. در این صورت:

$$totoweight = weight + \sum_{j=i+1}^{k-1} w_j$$

$$bound = \underbrace{(profit + \sum_{j=i+1}^{k-1} p_j)}_{\text{بهره حاصل از } k\text{-1 قطعه نخست}} + \underbrace{(W - totweight)}_{\text{ظرفیت باقیمانده برای قطعه } k \text{اُم}} * \underbrace{\frac{p_k}{w_k}}_{\text{بهره واحد وزن برای قطعه } k \text{اُم}}$$

- گره غیر امید بخش

$$bound \leq maxprofit$$

مثال ۵-۶

- $n = 4, \; W = 16$

i	p_i	w_i	p_i / w_i
1	\$40	2	\$20
2	\$30	5	\$6
3	\$50	10	\$5
4	\$10	5	\$2

مثال

$$maxprofit = 0$$

$$profit = 0$$

$$weight = 0$$

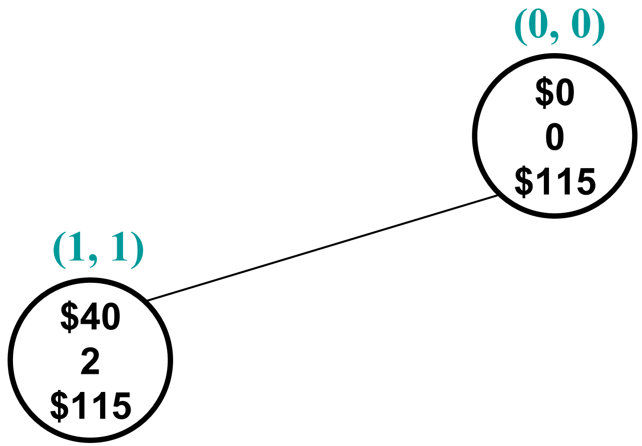
$$totoweight = weight + \sum_{j=0+1}^{3-1} w_j = 0 + 2 + 5 = 7$$

$$bound = (profit + \sum_{j=0+1}^{3-1} p_j) + (W - totweight) \; * \; \frac{p_3}{w_3}$$

$$= \$0 + \$40 + \$30 + (16 - 7) * \frac{\$50}{10} = \$115$$

این گره امید بخش می باشد زیرا وزنش کمتر از ۱۶ (W) و حدش بزرگتر از $maxprofit = 0$ می باشد.

مثال



Item 1

($40, 2)

$$profit = \$0 + \$40 = \$40$$

$$weight = 0 + 2 = 2$$

$$maxprofit = \$40$$

$$totoweight = weight + \sum_{j=1+1}^{3-1} w_j = 0 + 2 + 5 = 7$$

$$bound = (profit + \sum_{j=1+1}^{3-1} p_j) + (W - totweight) * \frac{p_3}{w_3}$$

$$= \$0 + \$40 + \$30 + (16 - 7) * \frac{\$50}{10} = \$115$$

این گره امید بخش می باشد زیرا وزنش کمتر از ۱۶ (W) و حدش بزرگتر از $maxprofit = 40$ می باشد.

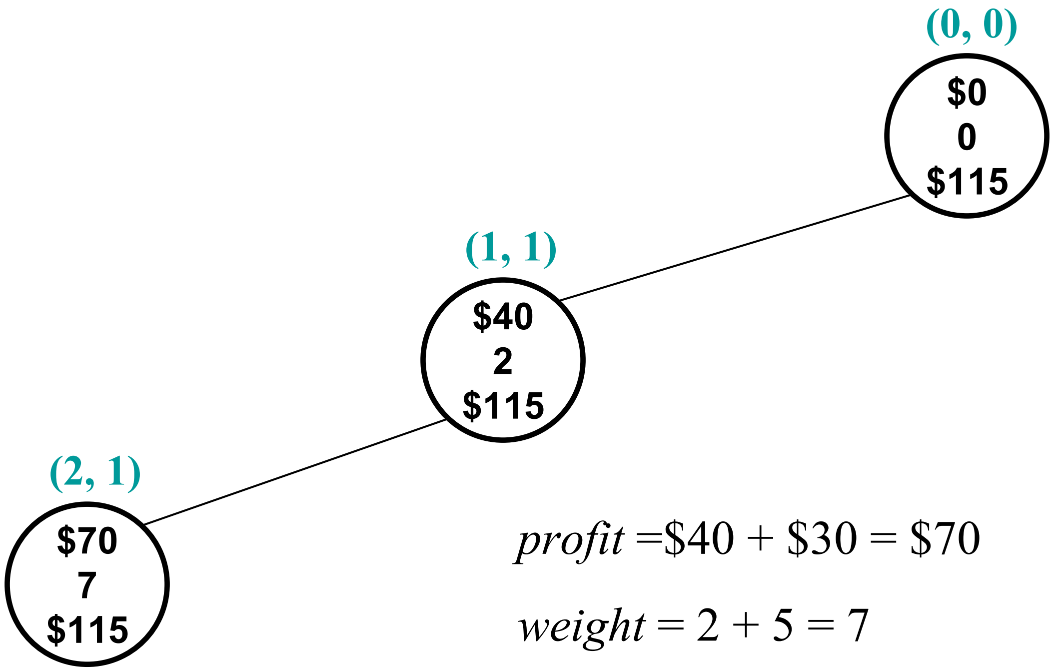
مثال

Item 1

($40, 2)

Item 2

($30, 5)



$$profit = \$40 + \$30 = \$70$$

$$weight = 2 + 5 = 7$$

$$maxprofit = \$70$$

$$totoweight = weight + \sum_{j=2+1}^{3-1} w_j = 7$$

$$bound = \$70 + (16 - 7) * \frac{\$50}{10} = \$115$$

این گره امید بخش می باشد زیرا وزنش کمتر از ۱۶ (W) و حدش بزرگتر از $maxprofit = 70$ می باشد.

مثال

Item 1

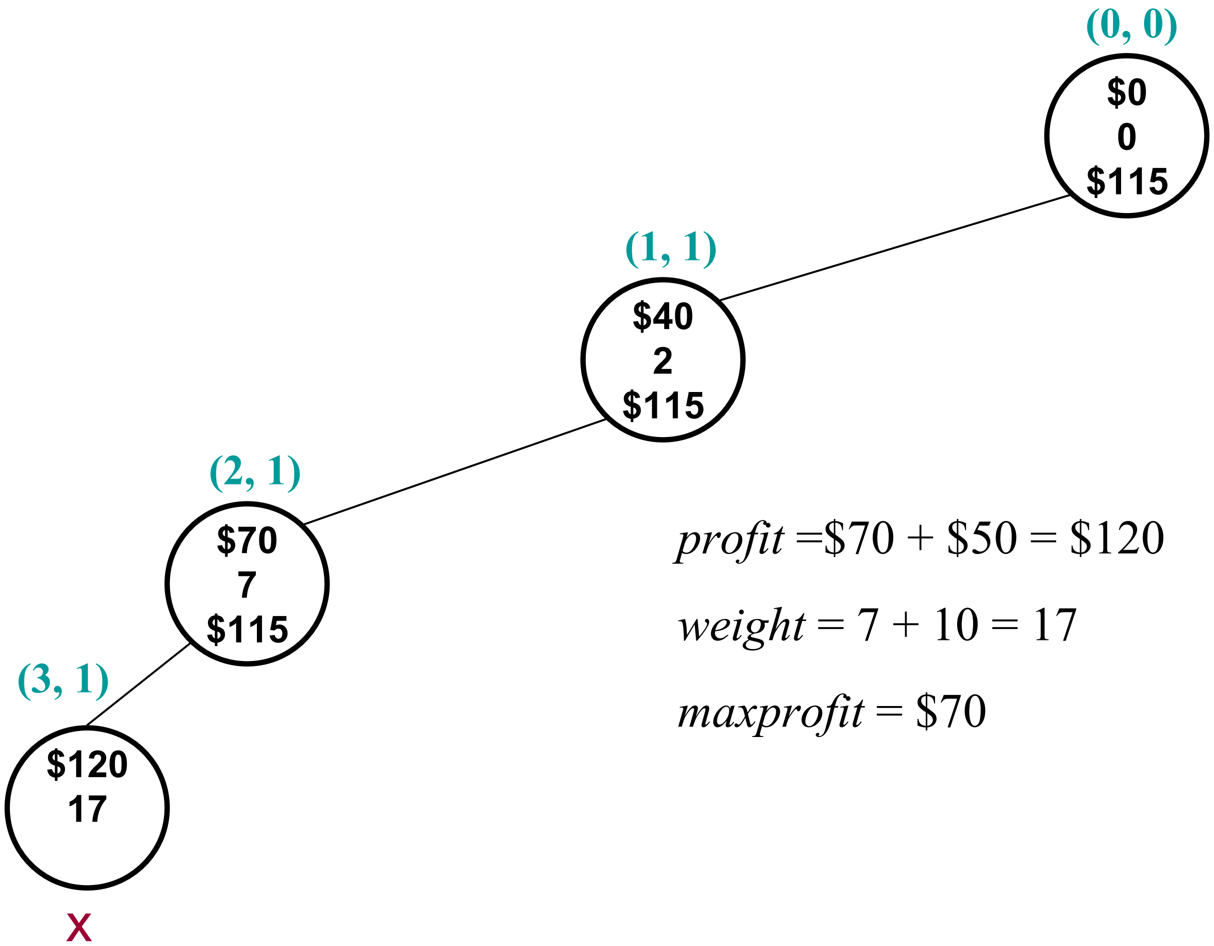
($40, 2)

Item 2

($30, 5)

Item 3

($50, 10)



$profit = \$70 + \$50 = \$120$

$weight = 7 + 10 = 17$

$maxprofit = \$70$

این گره امید بخش نمی باشد زیرا وزنش بیشتر از ۱۶ (W) است و بنابراین حدش را محاسبه نمی کنیم

مثال

Item 1

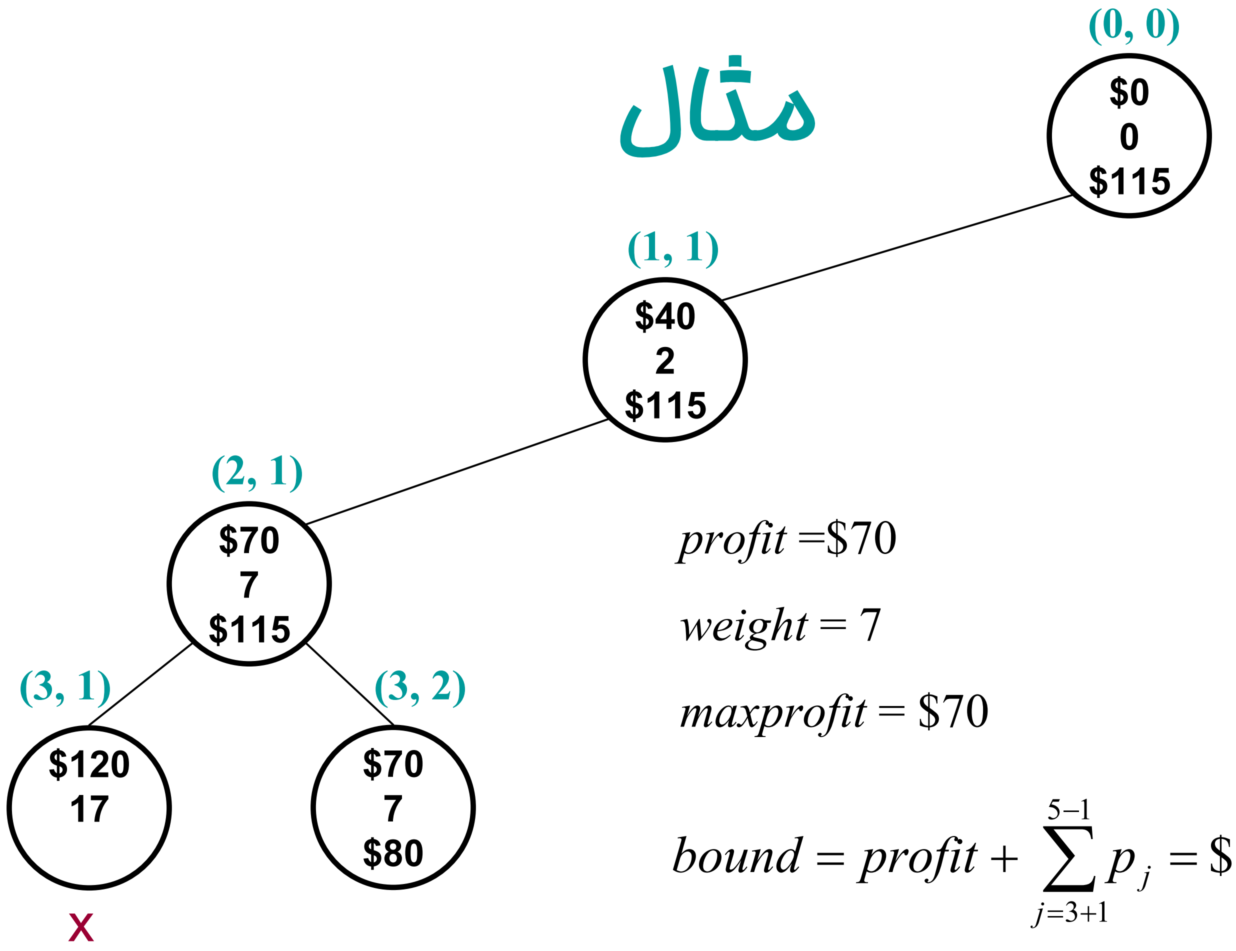
($40, 2)

Item 2

($30, 5)

Item 3

($50, 10)



$$profit = \$70$$

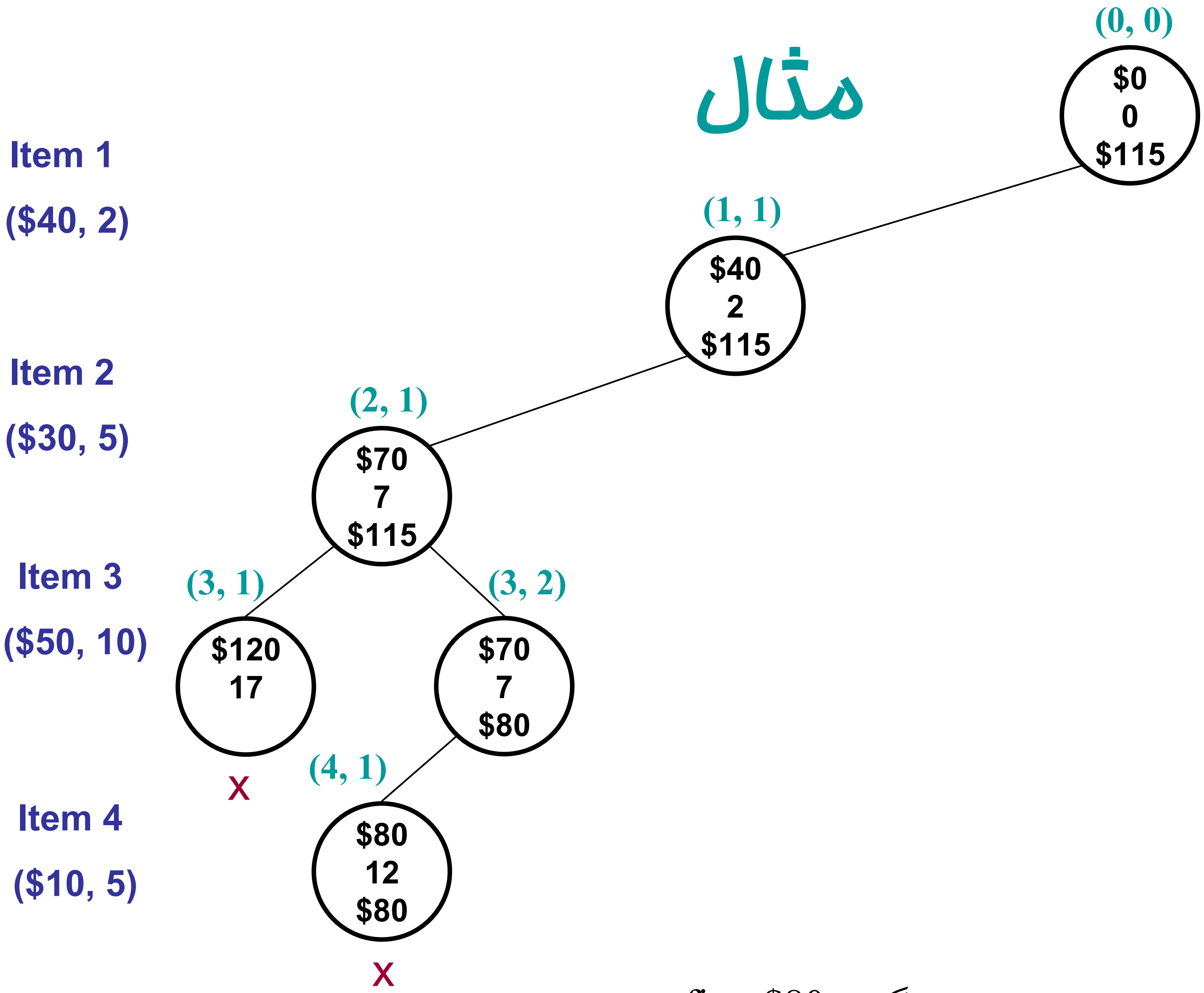
$$weight = 7$$

$$maxprofit = \$70$$

$$bound = profit + \sum_{j=3+1}^{5-1} p_j = \$70 + \$10 = \$80$$

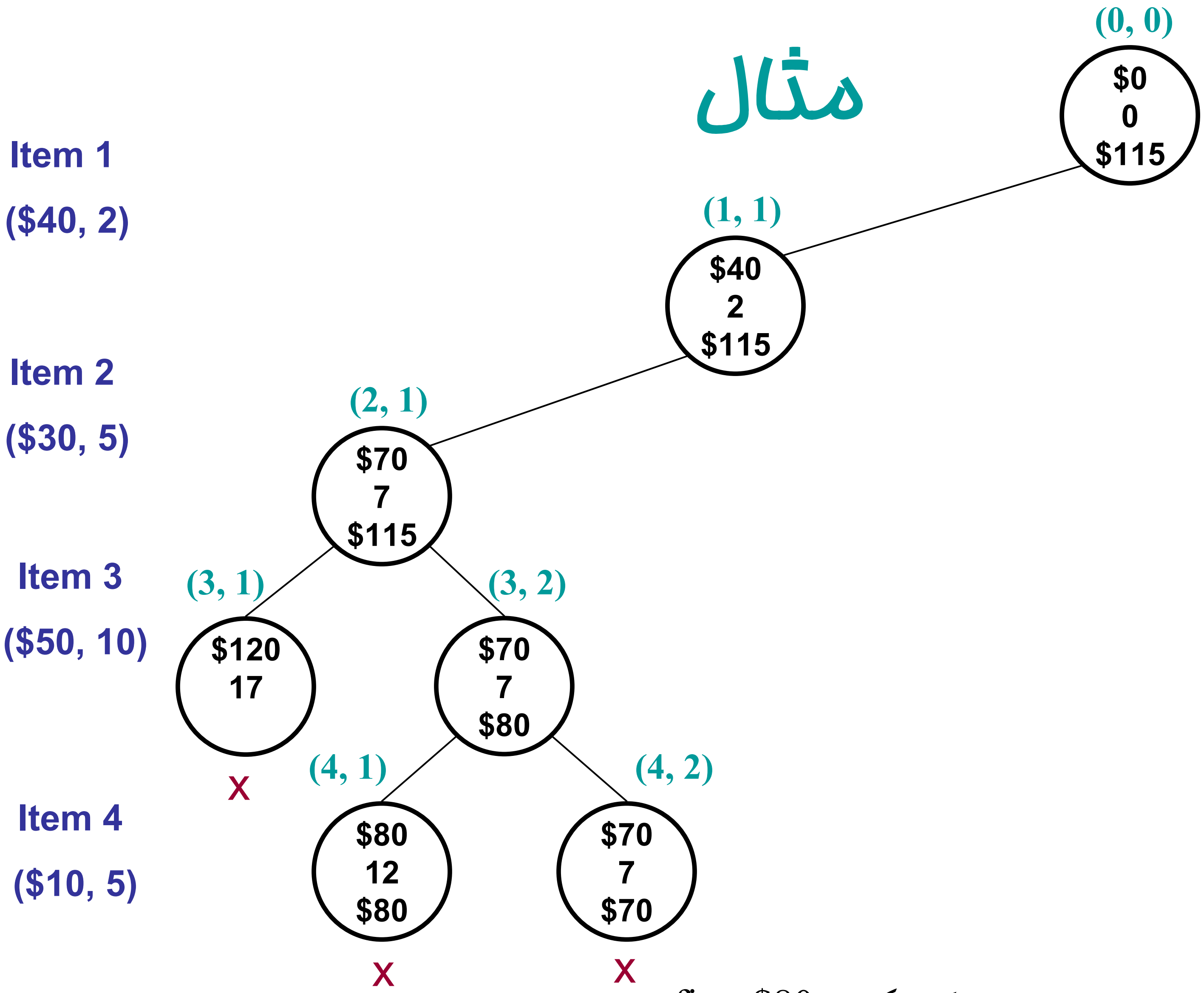
این گره امید بخش می باشد زیرا وزنش کمتر از ۱۶ (W) است و حدش بزرگتر از $maxprofit = \$70$ می باشد

مثال



این گره امید بخش نمی باشد زیرا حدش بزرگتر از maxprofit = $80 نمی باشد

مثال



این گره امید بخش می باشد زیرا حدش کوچکتر از $80 = maxprofit می باشد

مثال

Item 1

($40, 2)

Item 2

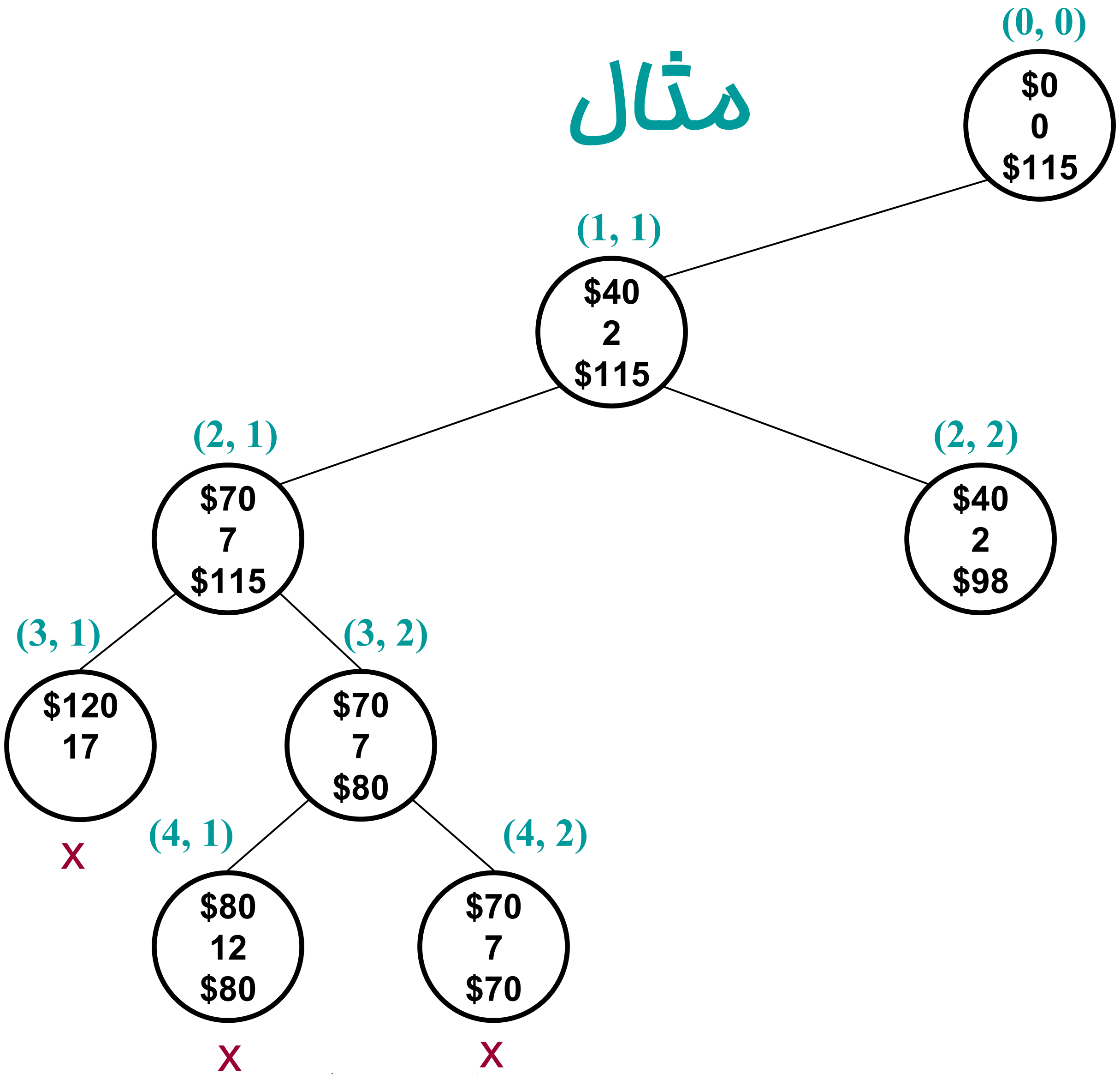
($30, 5)

Item 3

($50, 10)

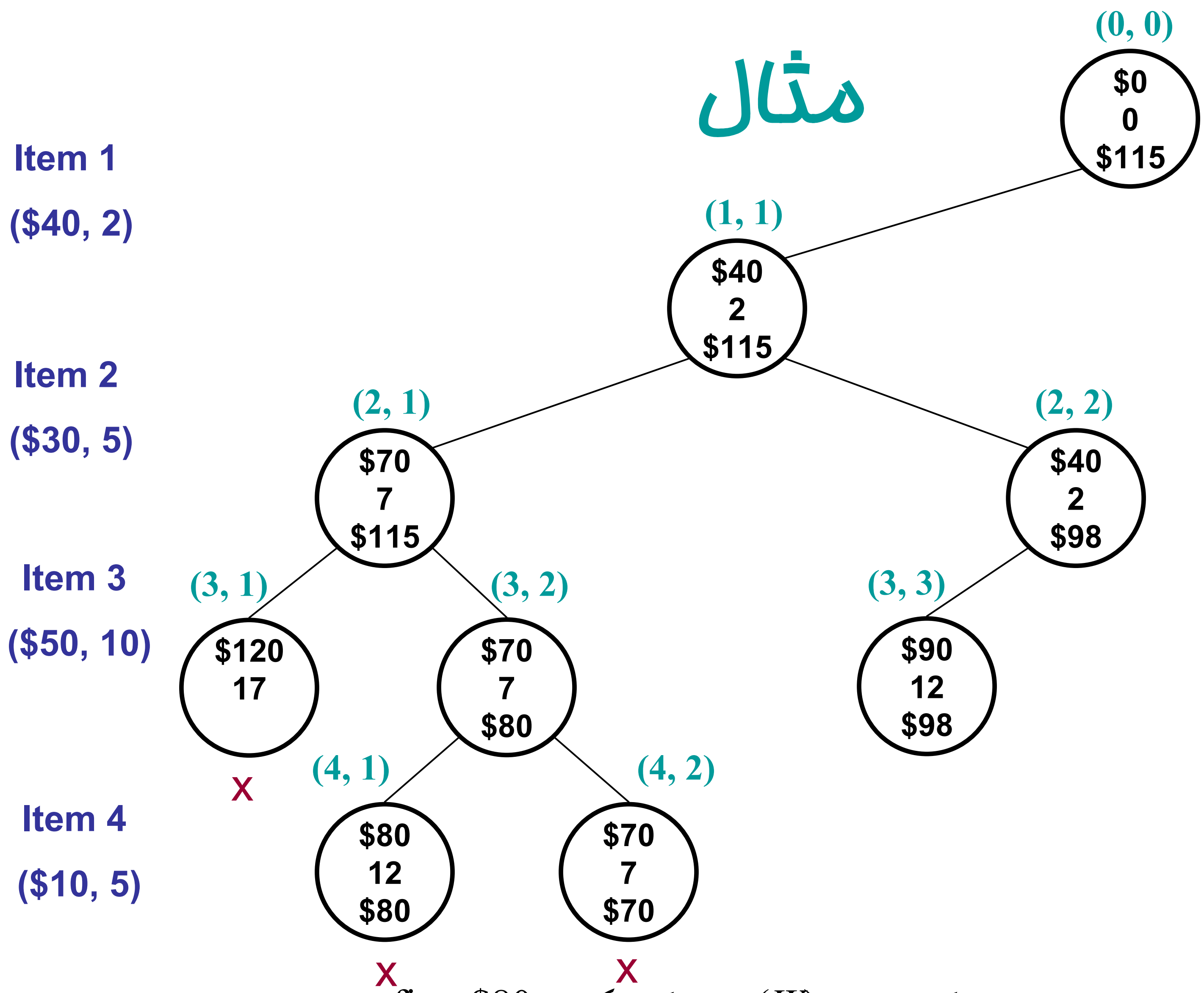
Item 4

($10, 5)



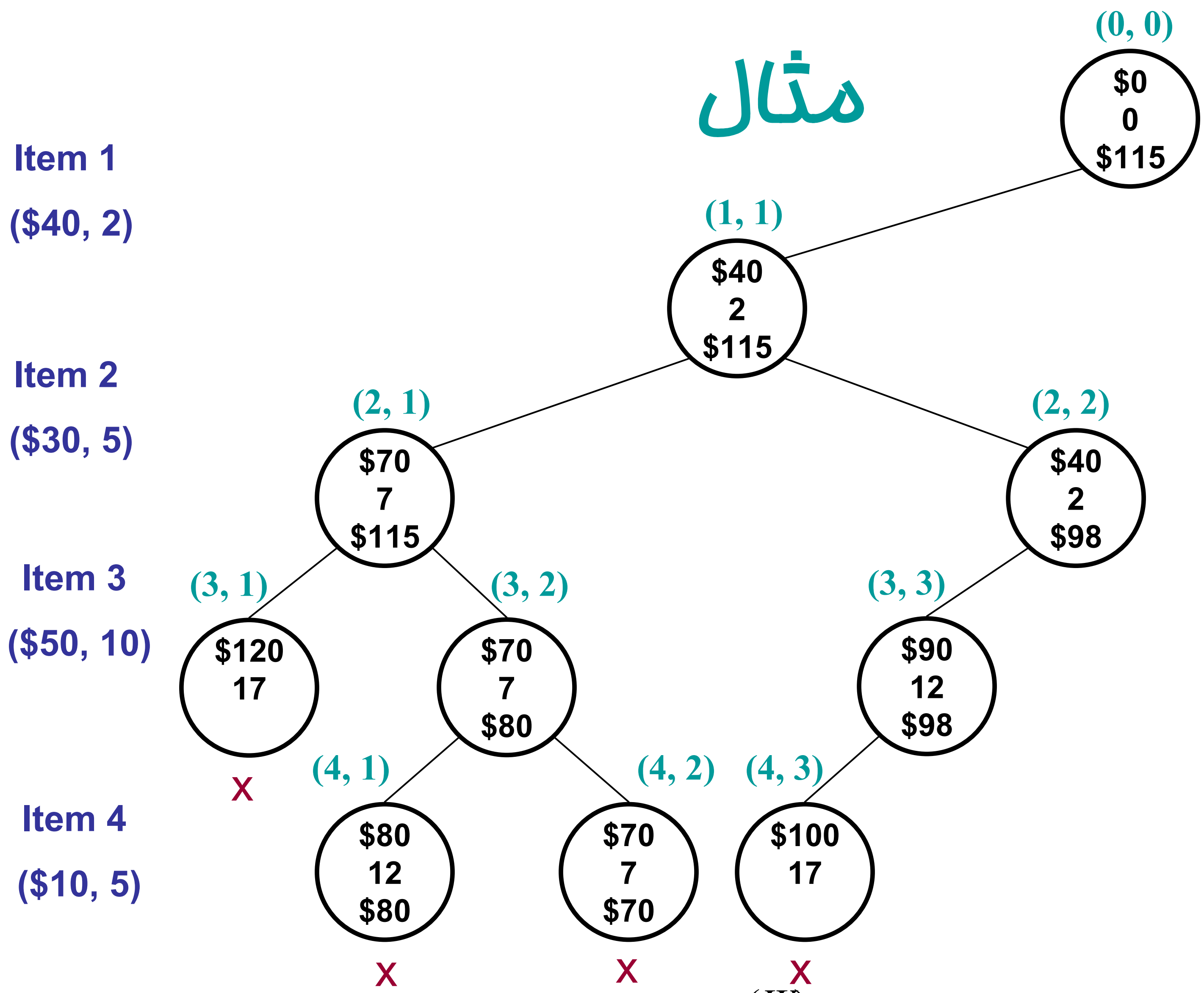
این گره امید بخش می باشد زیرا وزنش کمتر از ۱۶ (W)حدش کوچکتر از $\text{maxprofit} = \$80$ می باشد

مثال



این گره امید بخش می باشد زیرا وزنش کمتر از ۱۶ (W) حدش کوچکتر از $\text{maxprofit} = \$80$ می باشد

مثال



این گره امید بخش نمی باشد زیرا وزنش بیشتر از ۱۶ (W)و بنابراین حدش محاسبه نمی شود

مثال

Item 1

($40, 2)

Item 2

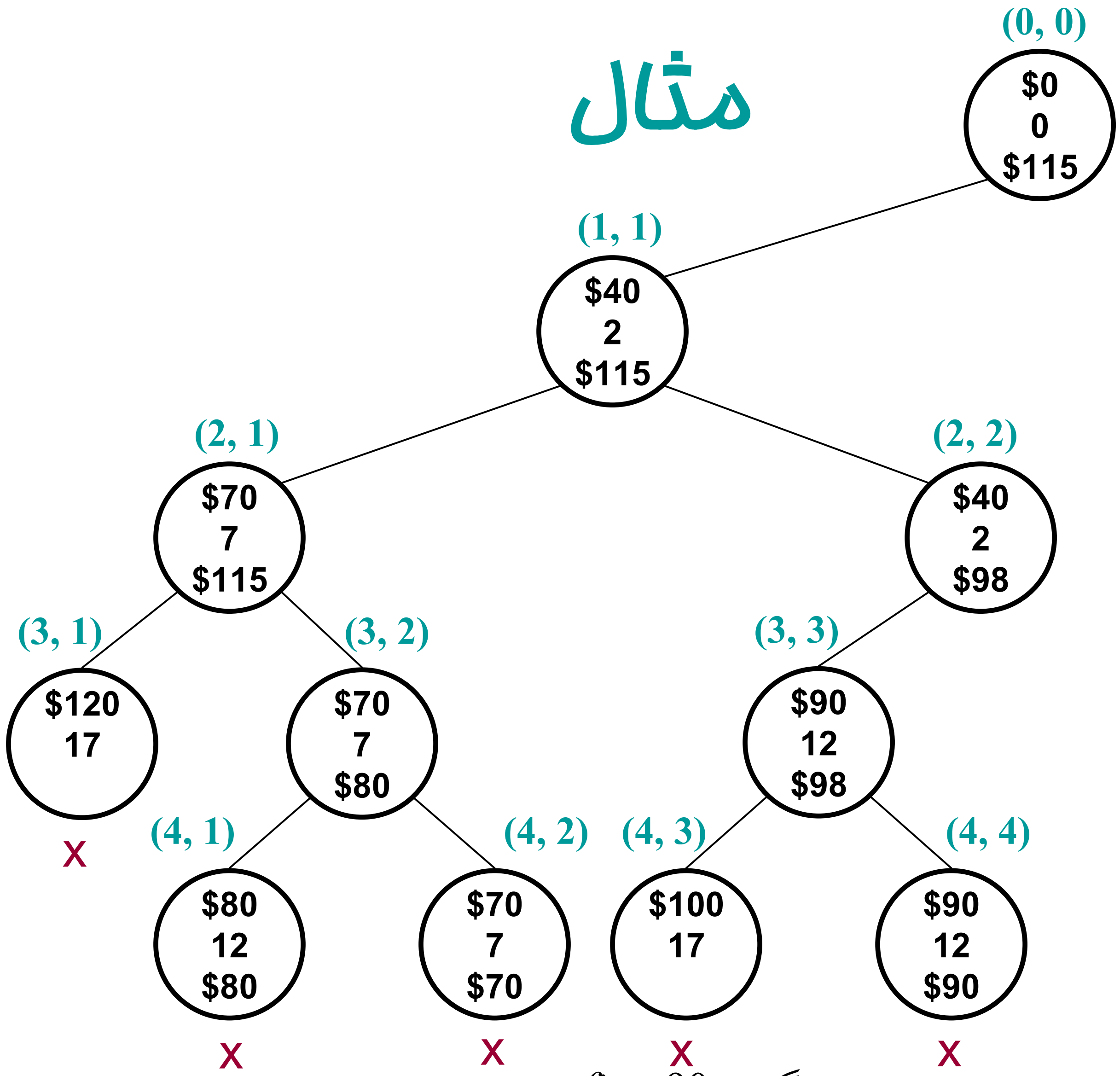
($30, 5)

Item 3

($50, 10)

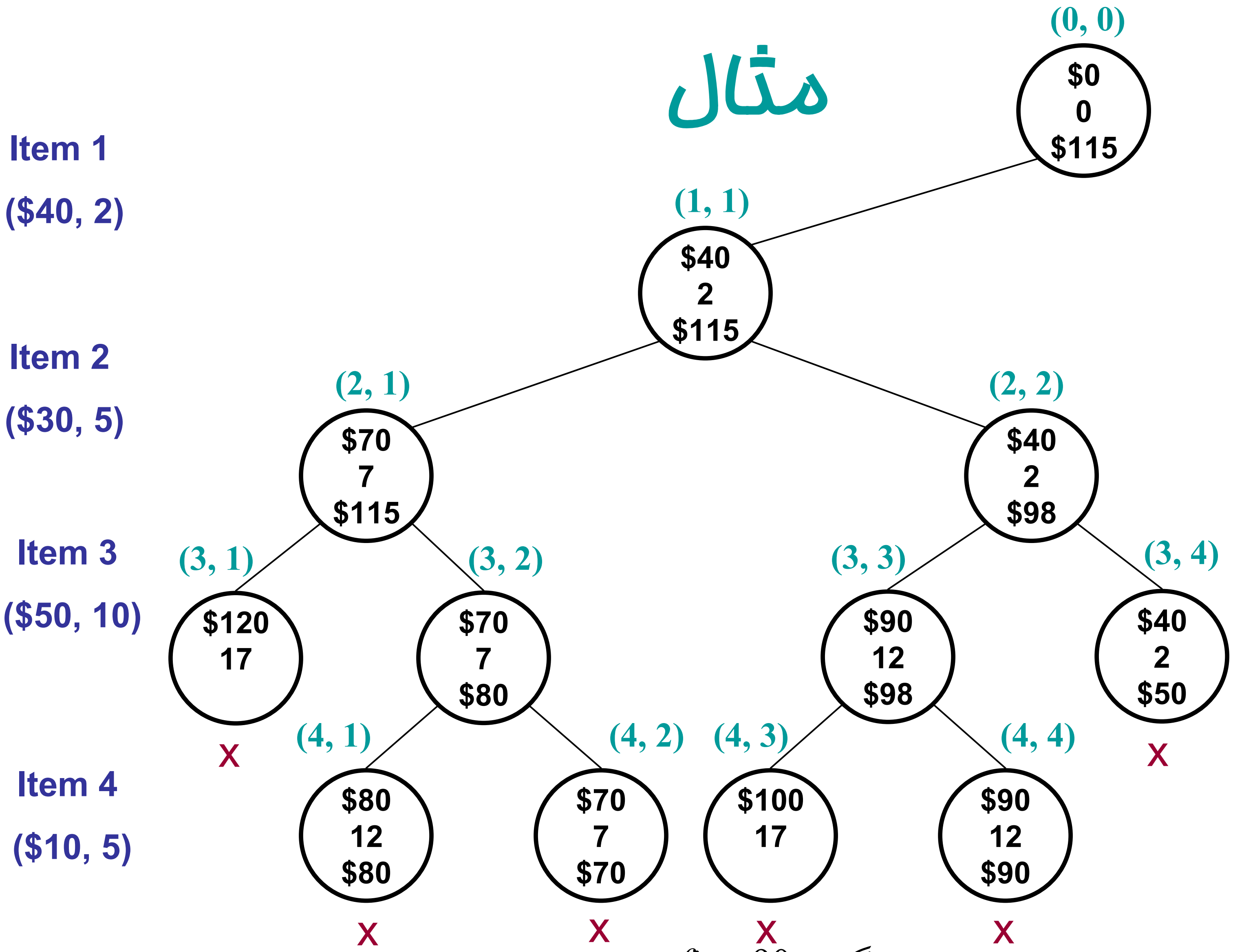
Item 4

($10, 5)



این گره امید بخش نمی باشد زیرا حدش بزرگتر از $maxprofit = 90$ نمی باشد

مثال



این گره امید بخش نمی باشد زیرا حدش بزرگتر از $maxprofit = 90$ نمی باشد

الگوریتم

- مساله: n قطعه که هر یک دارای وزن و ارزش مشخصی می باشد، داده شده است. وزن و ارزش هر قطعه یک عدد صحیح و مثبت می باشد. علاوه براین، عدد صحیح و مثبت W داده شده است. مطلوب است تعیین مجموعه ای از قطعات با حداکثر ارزش به شرط آن که حاصل جمع اوزان آنها از W بیشتر نباشد.

- ورودی ها: اعداد صحیح و مثبت n و W. آرایه های w و p که هر کدام از ۱ تا n اندیس گذاری شده اند و حاوی اعداد صحیح و مثبتی می باشند که که بر اساس مقادیر p_i / w_i به صورت غیر نزولی مرتب شده اند.

- خروجی ها: آرایه $bestset$ که از ۱ تا n اندیس گذاری شده است و در آن مقدار $bestset[i]$ در صورتی "yes" می باشد که قطعه i اُم در مجموعه بهینه گنجانده شود. عدد صحیح $maxprofit$ که ارزش بیشینه را نشان می دهد.

الگوریتم

```
void knapsack ( index i, int profit, int weight )
{
    if ( weight <= W && profit > maxprofit ) {
            maxprofit = profit ;
            numbest = i ;
            bestset = include ;
    }
    if ( promising (i)) {
            include [i + 1] = "yes" ;
            knapsack (i + 1, profit + p [i + 1], weight + w [i + 1] ) ;
            include [i + 1] = "no" ;
            knapsack ( i + 1, profit, weight) ;
    }
}
```

الگوریتم

```
bool promising ( index i)
{
    index j, k ;
    int totweight ;
    float bound ;

    if ( weight >= W)
            return false ;
    else {
            j = i + 1 ;
            bound = profit ;
            totweight = weight ;
            while ( j <= n && totweight + w [j] <= W) {
                        totweight = totweight + w [j] ;
                        bound = bound + p [j];
                        j++;
            }
            k = j;
            if ( k <= n)
                        bound = bound + ( W – totweight * p [k] / w [k] ;
            return bound > maxprofit ;
    }
}
```

پیچیدگی زمانی

- تعداد گره های درخت فضای حالت $2^{n+1}-1$
- مثال از بدترین حالت

$$p_i = 1 \qquad w_i = 1 \qquad 1 \le i \le n-1$$

$$p_n = n \qquad w_n = n$$

مقایسه الگوریتم برنامه ریزی پویا و عقبگرد برای مساله کوله پشتی ۰-۱

- الگوریتم برنامه نویسی پویا در بدترین حالت

$$O(minimum(2^n, nW))$$

- الگوریتم عقبگرد

$$\Theta(2^n)$$

- هوروویتز و ساهنی نشان داده اند که الگوریتم عقبگرد معمولا نسبت به الگوریتم برنامه ریزی پویا کارآیی بیشتری دارد.
- ترکیب روش تقسیم و حل و برنامه ریزی پویا توسط هوروویتز و ساهنی برای حل مساله کوله پشتی ۰-۱
  - در بدترین حالت $O(2^{n/2})$
  - این الگوریتم معمولا کارآیی بیشتری نسبت به عقب گرد دارد.