دانشکده مهندسی کامپیوتر

جزوه درس

ساختمان‌های داده

استاد درس: سید صالح اعتمادی

پاییز ۱۳۹۸

جلسه ۱۲

برنامه نویسی پویا-ادامه

یاسین عسکریان - ۱۳۹۸/۸/۶

۱.۱۲ کوله پشتی [۱]

مسئله کوله‌پشتی که با نام‌های Knapsack یا Rucksack مطرح می‌شود فرض کنید مجموعه‌ای از اشیا که هر کدام داری وزن و ارزش خاصی هستند در اختیار دارید. به هر شی تعدادی را تخصیص دهید به‌طوری‌که وزن اشیا انتخاب شده کوچکتر یا مساوی حدی از پیش تعیین شده، و ارزش آن‌ها بیشینه شود. علت نامگذاری این مسئله، جهانگردی است که کوله‌پشتی‌ای با اندازهٔ محدود دارد و باید آن را با مفیدترین صورت ممکن از اشیا پر کند. [۱]
از مثال های کاربردی دیگر آن می توان به تبلیغات تلویزیونی اشاره کرد فرض کنید شما مسئول تبیلغات یک شبکه ی تلویزیونی هستید،که قصد دارید تبلیغات را در حین پخش یک سریال پرطرفدار پخش کنید خب با توجه به زمانی که در اختیار دارید مثلا ۵ دقیقه باید از مجموعه ای از پیشنهادات تبلیغاتی گزینه هایی را انتخاب کنید تا بیشترین سود را برای شبکه داشته باشد
این نوع مسئله به دو حالت کلی تبدیل می شود که در ادامه به بررسی آن ها خواهیم پرداخت.

۱.۱.۱۲ کوله پشتی کسری [۲]

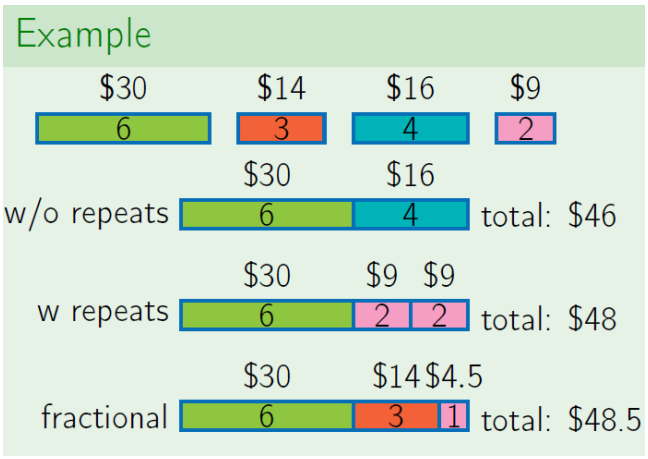
فرض کنید شما یک کوله پشتی دارید و می خواهید از یک خواربار فروشی اجناسی را انتخاب کنید که مجموع حجم شان از حجم کوله پشتی شما کمتر یا مساوی آن باشد و مجموع ارزش این اجناس در بیشترین حالت ممکن باشد خب مثلا اگر یک کیلو زعفران به ارزش صد هزار تومان نیم کیلو دارچین به ارزش هفتاد هزار تومان دارید و حجم کوله پشتی شما سیصد گرم باشد شما می توانید **کسری** از هرکدام از اجناس را انتخاب و قیمت آن ها را محاسبه کنید تا مجموع قیمت آن بیشینه شود
خب برای حل این مسئله از الگوریتم حریصانه [۳] استفاده می کنیم که در جلسه چهارم به بحث و بررسی آن پرداختیم.

---

[۱] Knapsack
[۲] Fractional Knapsack
[۳] Greedy Algorithms

۲.۱۲ کوله پشتی گسسته [۴]

فرض کنید شما یک کوله پشتی با حجم مشخص دارید و میخواهید آن رو با شمش طلا،نقره و برنز پر کنید تا مجموع ارزش آن بیشترین حالت ممکن باشد می دانیم که نمی توانیم کسری از یک شمش طلا را برداریم پس یا باید یک شمش طلا به کوله پشتی اضافه کنیم یا اصلا آن را اضافه نکنیم
این مسئله به دو حالت با تکرار و بدون تکرار تبدیل می شود در حالت با تکرار با تو جه به مثال بالا ما می توانیم چند شمش طلا برداریم ولی در حالت بدون تکرار حداکثر یک شمش طلا. لطفا به شکل زیر توجه کنید :

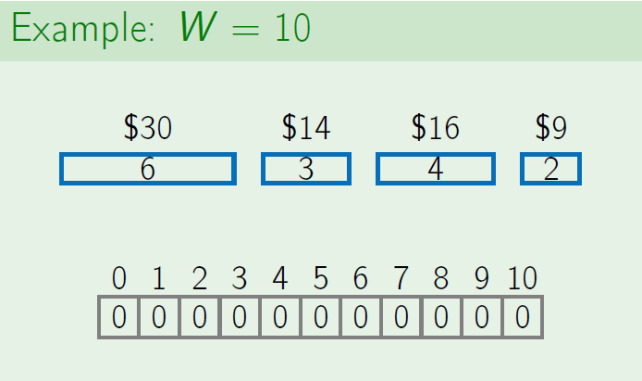


شکل ۱.۱۲: مقایسه حالت های مختلف مسئله کوله پشتی

۱.۲.۱۲ کوله پشتی با تکرار [۵]

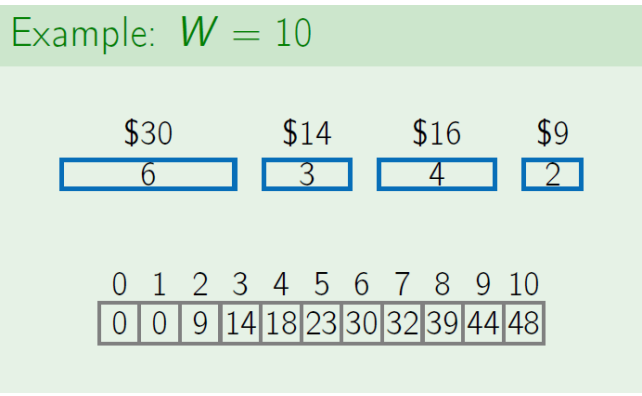
برای حل این مسئله باید ابتدا بیشینه ی ارزش برای حجم ها کم تر از حجم کوله پشتی خودمان را بدست آوریم به مثال زیر توجه کنید

---
[۴] Discrete Knapsack
[۵] Knapsack with Repetitions



شکل ۲.۱۲: کوله پشتی با تکرار(۱)

خب ما چهار جنس مختلف با حجم و ارزش های متفات داریم و حجم کوله پشتی ما نیز ده است خب ابتدا برای حجم صفر شروع به محاسبه بیشنه ارزش میکنیم و با توجه به اجناس در حجم صفر بیشینه نیز صفر است و همینطور برای حجم یک، خب برای حجم دو ما یک جنس به ارزش ۹ دلار و حجم دو داریم و جنس دیگری را نمی توانیم در داخل آن قرار دهیم پس بیشینه ی ارزش در حجم دو برابر با ۹ دلار می شود حال به سراغ حجم سه می رویم در این حجم ما دو جنس به حجم های دو و سه داریم خب اگر حجم دو را انتخاب کنیم ۹ دلار به ارزش آن اضافه شده و دو واحد از حجم آن کاسته می شود خب یک واحد حجم باقی می ماند برای پر کردن آن حجم به بیشینه ارزش در حجم یک رجوع کرده و مقدار آن که برابر با صفر هست را با ۹ دلار جمع می کنیم حال در حالت بعدی جنسی که حجم آن سه و ارزش آن ۱۴ دلار هست را انتخاب کرده که در اینصورت حجم باقی مانده برابر صفر می شود حال ارزش به دست آمده در این دو حالت را با هم مقایسه کرده و ارزش بیشتر را برای حجم سه انتخاب می کنیم که برابر است با ۱۴ دلار و به همین ترتیب ارزش ها را تا حجم ۱۰ حساب می کنیم



شکل ۳.۱۲: کوله پشتی با تکرار(۲)

در اینصورت الگوریتم حل مسئله کوله پشتی با توانایی تکرار اجناس به صورت زیر است

```
Knapsack(W)
value(0) ← 0
for w from 1 to W:
  value(w) ← 0
  for i from 1 to n:
    if w_i ≤ w:
      val ← value(w − w_i) + v_i
      if val > value(w):
        value(w) ← val
return value(W)
```

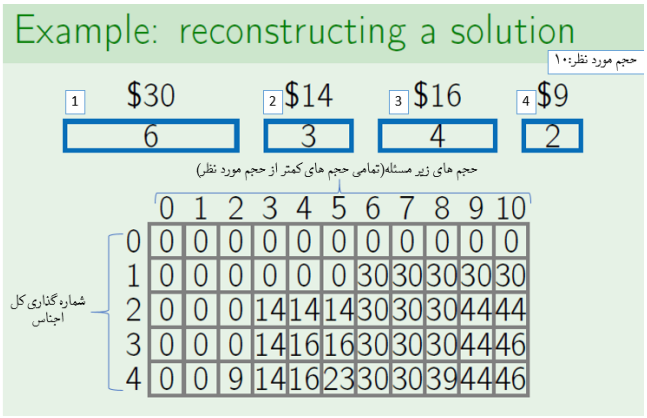
شکل ۴.۱۲: الگوریتم کوله پشتی با تکرار

حال اگر بخواهیم بدانیم که چه اجناسی را انتخاب کردیم تا به این بیشینه رسیدیم و ما آرایه زیرمسئله،حجم های کوچک تر از حجم مورد نظر،را داریم از اخرین خانه ی ارایه شروع میکنیم ابتدا حجم تمام اجناس را به ترتیب از حجم خانه ی آخر کم می کنیم خب تفاضل بدست آمده خود نیز یکی از زیر مسئله های ماست پس به سراغ خانه ای با شماره ی تفاضل بدست آمده میرویم اگر جنسی که ما انتخاب کردیم یکی از اجناسی باشد که در کوله پشتی قرار گرفته است باید جمع ارزش آن جنس با ارزش قرار گرفته در خانه ی بدست آمده برابر با ارزش نهایی شده باشد در غیراینصورت به سراغ جنس بعدی میرویم اگر جنس انتخابی درست باشد همین مراحل را برای آن خانه ی بدست آمده در آرایه تکرار میکنیم و آنقدر تکرار میکنیم تا به خانه ای با ارزش صفر برسیم

۲.۲.۱۲ کوله پشتی بدون تکرار [۶]

در این حالت زیرمسئله تنها حجم های کوچک تر از حجم مورد نظر نیست بلکه با این دید به مسئله نگاه می کنیم که پر کردن تمامی حجم ها ابتدا با جنس شماره یک سپس شماره یک و دو و همینطور الی آخر انجام شود. پس ما به یک آرایه دو بعدی نیاز داریم به شکل زیر توجه کنید

[۶] Knapsack without Repetitions



شکل ۵.۱۲: بازسازی راه حل کوله پشتی بدون تکرار

با تو به شکل بالا اگر اسم آرایه دو بعدی M باشد خانه ی M(i,j) بیانگر آن است که کوله پشتی ای با حجم i و با استفاده از اجناس با شماره ی کوچک تر مساوی j یعنی [j,j-1,j-2,j-3,...,0] حداکثر M(i,j) ارزش در آن جای گرفته است
و الگوریتم آن را نیز می توانید در شکل زیر مشاهده کنید

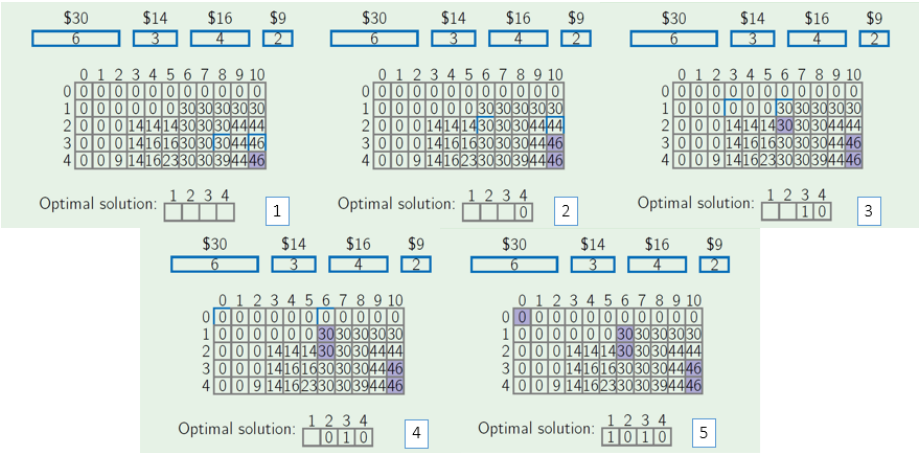
Knapsack(W)

```
initialize all value(0, j) ← 0
initialize all value(w, 0) ← 0
for i from 1 to n:
  for w from 1 to W:
    value(w, i) ← value(w, i − 1)
    if w_i ≤ w:
      val ← value(w − w_i, i − 1) + v_i
      if value(w, i) < val
        value(w, i) ← val
return value(W, n)
```

شکل ۶.۱۲: الگوریتم کوله پشتی بدون تکرار

حال اگر ما آن آرایه دو بعدی را داشته باشیم و بخواهیم بدانیم چه اجناسی را انتخاب کرده ایم ابتدا به آخرین خانه یعنی M(imax,jmax) مراجعه می کنیم این خانه بیانگر آن است که آیا ما از جنس jmax استفاده کرده ایم یا خیر اگر استفاده کرده باشیم باید جمع ارزش خانه ای با حجم (W-w(jmax)) با value(jmax)

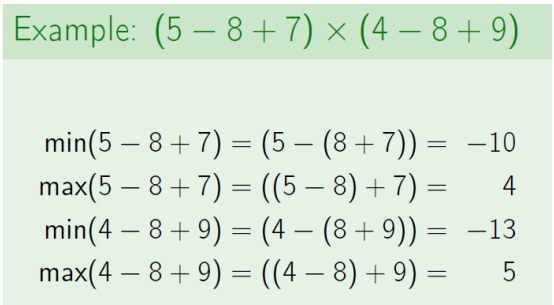
برابر با M(imax,jmax) شود در غیر اینصورت یعنی از جنس با شماره jmax استفاده نکرده ایم برای درک بیشتر به شکل زیر توجه کنید



شکل ۷.۱۲: کوله پشتی بدون تکرار

۳.۱۲ قرار دادن پرانتز [۷]

در این مسئله ورودی های ما متشکل از یک دنباله اعداد و یک دنباله از چهار عمل اصلی ریاضی است به این که بین هردو عدد حتما یک عملگر وجود دارد و ما وظیفه داریم تا با پرانتز گذاری مناسب بیشینه یا کمینه ی جواب نهایی این عبارت را بدست آوریم
فرض کنید عبارتی مثل $5 - 8 + 7 \times 4 - 8 + 9$ و می خواهیم بیشینه این عبارت را در حالت زیر بدست آوریم
$(5 - 8 + 7) \times (4 - 8 + 9)$



شکل ۸.۱۲: بررسی مثال برای مسئله قرار دادن پرانتز

[۷] Parentheses Placing

با توجه به شکل بالا در می یابیم که بیشینه عبارت $(9 + 8 - 4) \times (7 + 8 - 5)$ برابر است با ضرب دو عدد منفی یعنی (-13) × (-10) که برابر است با 130 خب همانطور که مشاهده می کنید ما برای بدست آوردن بیشینه این عبارت ابتدا باید کمینه دو بخش از آن را بدست آوریم پس همانطور که از این مثال متوجه شدید برای بدست آوردن بیشینه عبارت ما باید کمینه و بیشینه بخش های مختلف را بدست آوریم
اگر E یک زیرعبارت [۸] به صورت

$$E = d(i)op(i)...d(j)op(j)$$

در این صورت داریم
M(i,j)= maximum value of E
m(i,j)= minimum value of E
و رابطه بازگشتی آن ها به صورت زیر است

$$M(i,j) = \max_{i \le k \le j-1} \begin{cases} M(i,k) & op_k & M(k+1,j) \\ M(i,k) & op_k & m(k+1,j) \\ m(i,k) & op_k & M(k+1,j) \\ m(i,k) & op_k & m(k+1,j) \end{cases}$$

$$m(i,j) = \min_{i \le k \le j-1} \begin{cases} M(i,k) & op_k & M(k+1,j) \\ M(i,k) & op_k & m(k+1,j) \\ m(i,k) & op_k & M(k+1,j) \\ m(i,k) & op_k & m(k+1,j) \end{cases}$$

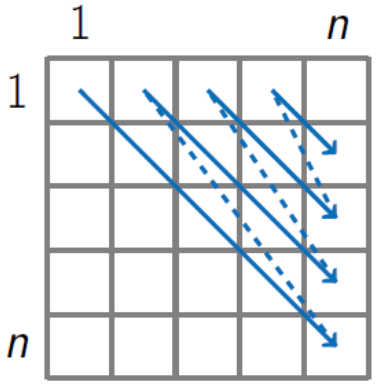
پس ما به دو آرایه دو بعدی یکی برای کمینه و دیگری برای بیشنه نیاز داریم در نهایت الگوریتم محاسبه کمینه و بیشیته هر زیرعبارت به صورت زیر است

```
MinAndMax(i, j)
min ← +∞
max ← −∞
for k from i to j − 1:
    a ← M(i, k)  op_k  M(k + 1, j)
    b ← M(i, k)  op_k  m(k + 1, j)
    c ← m(i, k)  op_k  M(k + 1, j)
    d ← m(i, k)  op_k  m(k + 1, j)
    min ← min(min, a, b, c, d)
    max ← max(max, a, b, c, d)
return (min, max)
```

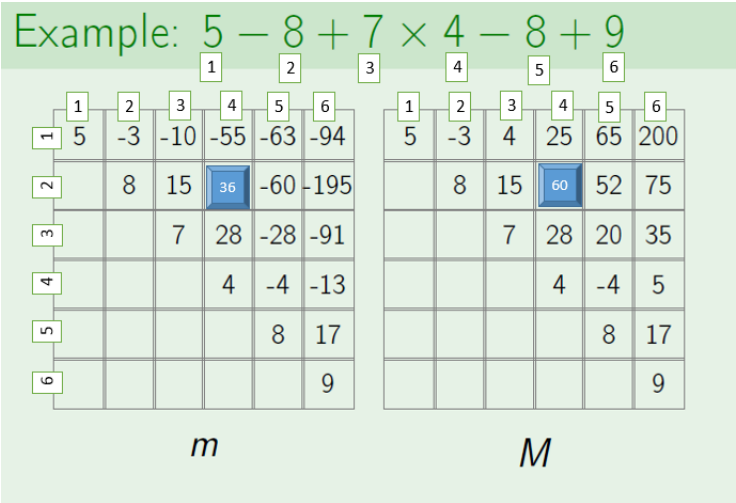
شکل ۹.۱۲: الگوریتم محاسبه کمینه و بیشینه

[۸] subexpression

خب برای حل مسئله ابتدا به سراغ ساده ترین زیرعبارت می رویم یعنی هر عدد بدون هیچ عملگری (i=j) در این زیرعبارت ها حاصل هر زیر عبارت برابر است با خود زیرعبارت یعنی خود عدد. زیرعبارت بعدی عبارتی است با دو عدد و یک عملگر بین آن ها (|j-i|=1) حالت بعدی برابر است با سه عدد و دو عملگر بین آنها (|j-i|=2) در این حالت این زیر عبارت مه به دو بخش به صورتی که یک بخش که شامل دو عدد و یک عملگر است با یک عملگر به بخش دیگر که یک عدد است که این دو بخش را نیز قبلا حل کرده ایم و همینطور الی آخر با توجه به این تعاریف آرایه های دوبعدی ما به صورت زیر خواهد بود



حال به بررسی یک مثال می پردازیم



خب دو خانه مشخص شده در آرایه های دوبعدی کمینه و بیشینه یعنی M(2,4) و m(2,4) را مشاهده می کنید این خانه ها بیانگر زیرعبارت 8+7×4 یعنی از عدد شماره دو تا عدد شماره چهار خب این زیرعبارت

شامل دو بخش می باشد که به صورت $8+(7\times4)$ یا $(8+7)\times4$ می باشد که در حالت اول حاصل $8+7$ را قبلا محاسبه کرده ایم که برابر است با ۱۵ در خانه های $\mathrm{M}(2,3)$ و $\mathrm{m}(2,3)$ و حاصل این عبارت را در عدد ۴ ضرب می کنیم که این عدد هم در جدول ما هست خانه های $\mathrm{M}(4,4)$ و $\mathrm{m}(4,4)$ پس حاصل زیرعبارت اول ما برابر با 15×4 یعنی ۶۰ می شود حال به سراغ حل زیر عبارت بعد می رویم زیرعبارت (7×4) که بیشینه و کمینه این عبارت را می توانیم در خانه های $\mathrm{M}(3,4)$ و $\mathrm{m}(3,4)$ پیدا کنیم که این دو حالت کمینه و بیشینه باهم برابر و مساوی ۲۸ می باشند و حاصل این عبارت با ۸ جمع شده و برابر با ۳۶ می باشد پس مقایسه این دو در می یابیم که خانه $\mathrm{M}(2,4)=60$ و $\mathrm{m}(2,4)=36$ خواهد بود در نهایت الگوریتم حل مسئله پرانتزگذاری به صورت زیر است

```
Parentheses(d_1 op_1 d_2 op_2 ... d_n)
for i from 1 to n:
  m(i,i) ← d_i ,  M(i,i) ← d_i
for s from 1 to n − 1:
  for i from 1 to n − s:
    j ← i + s
    m(i,j), M(i,j) ← MinAndMax(i,j)
return M(1,n)
```

شکل ۱۰.۱۲: الگوریتم پرانتزگذاری

Bibliography

[1] "Knapsack." https://fa.wikipedia.org/wiki/%D9%85%D8%B3%D8%A6%D9%84%D9%87_%DA%A9%D9%88%D9%84%D9%87%E2%80%8C%D9%BE%D8%B4%D8%AA%DB%8C. Accessed: 2020-01-23.