فصل پنجم: برنامه‌نویسی پویا (جلسه اول)

مرکز آموزش الکترونیکی
دانشگاه علم و صنعت ایران

اهداف یادگیری

* معرفی اصل بهینه سازی
* آشنایی با روش برنامه‌نویسی پویا
* حل مسائلی نظیر:
- کوله پشتی ۱/۰
- گرافهای چند مرحله‌ای
- الگوریتم فلوید-وارشال
- مساله فروشنده دوره گرد

مقدمه

* برنامه‌نویسی پویا یکی از روشهای طراحی الگوریتم‌هاست که مانند روش حریصانه در حل مسائل بهینه سازی کاربرد دارد.
* در این روش در هر مرحله تصمیمی گرفته می‌شود، و نتیجه این تصمیمات منجر به حل مسئله می‌شود.
* در هر مرحله از تصمیم گیری، از نتایج مراحل قبل استفاده می شود، به طوریکه در نهایت یک رشته بهینه از تصمیمات تولید می‌شود.

مقدمه (ادامه)

* مثال: مساله کوله پشتی
* مثال: الگوهای ادغام بهینه
* مثال: کوتاهترین مسیر
* اصل بهینه سازی: بنابر اصل بهینه سازی، یک رشته بهینه از تصمیمات دارای خاصیت زیر است: حالت و تصمیم گیری مرحله اول هرچه که باشد، تصمیمات باقیمانده، با توجه به این حالت و تصمیم گیری، باید یک رشته بهینه تشکیل دهد.

مثال

* الگوریتم فیبوناچی:

```
int A[n];
A[0]=A[1]=1;
for(i=2;i<n;i++)
   A[i]=A[i-1]+A[i-2];
```

در اکثر مسائلی که با روش پویا حل می‌شوند، جهت نگهداری پاسخها در هر مرحله نیاز است تا از یک آرایه یک بعدی یا دو بعدی استفاده شود. ضمناً قبل از ارائه الگوریتم، باید بتوان مساله را بر اساس پاسخ بدست آمده در مراحال مختلف فرموله کرد. مثال: f(i)=f(i-1)+f(i-2)

کوله پشتی ۱/۰

* مساله کوله پشتی ۱/۰، مشابه مساله کوله پشتی است که قبلاً در فصل حریصانه بحث شد، با این تفاوت که x_iها می‌توانند فقط یکی از دو مقدار صفر یا یک را به خود بگیرند.

* هدف ماکزیمم سازی $\sum_{i=1}^{n} p_i x_i$ که در آن xi=0 or 1 است مشروط بر اینکه:

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

* مثال: W(4,6,2,7)، P(6,2,5,8) و M=5

راه حل حریصانه: X=(0,0,1,0)، سود = ۵

راه حل صحیح: X=(1,0,0,0)، سود = ۶

کوله پشتی ۱/۰ (نگرش پردازه‌ای)

* **ایده:** فرض کنید $g_i(y)$ مقدار سود ماکزیمم در مرحله iام است و y حجم باقی مانده از کوله پشتی باشد.
* در مرحله شروع باید $g_0(M)$ را محاسبه کرد.
* در هر مرحله تصمیم گیری در مورد انتخاب جسم iام صورت می گیرد. یعنی اگر آنرا انتخاب کنیم سود P_i بدست آمده و حجم باقی مانده $y-w_i$ می‌باشد. اما اگر آنرا انتخاب نکنیم سود حاصل برابر $g_{i+1}(y)$ می‌باشد.

کوله پشتی ۰/۱ (نگرش پردازه‌ای)

* ایده ذکر شده را می‌توان به صورت زیر فرموله کرد:

$$g_i(y)=\text{Max}\{g_{i+1}(y), g_{i+1}(y-w_i)+P_i\}$$

* با توجه به فرمول فوق: $g_n(y)=0$ و اگر $y<0$ آنگاه $g_i(y)=-\infty$

* چنانکه دیده می‌شود، تصمیم گیری در مرحله iام به تصمیمات آینده بستگی دارد به طوریکه رشته تصمیمات باید یک رشته بهینه ایجاد نماید. (با توجه به اصل بهینه سازی)

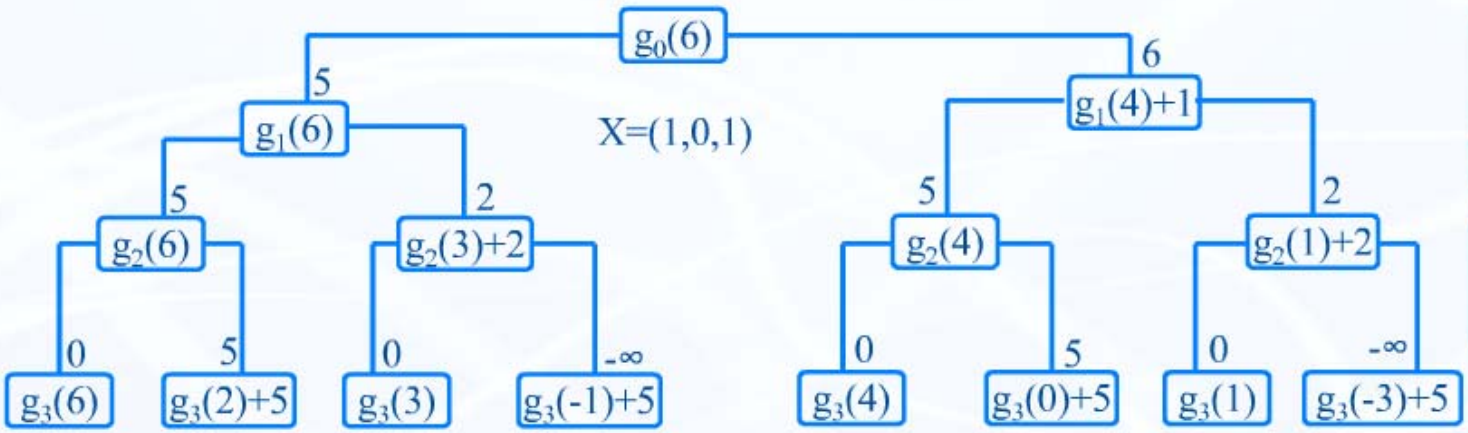
کوله پشتی ۱/۰ (نگرش پردازه‌ای)

* مثال: مراحل مختلف اجرای $g_0(M)$ را برای مساله کوله پشتی زیر نوشته، بردار جواب و مقدار جواب بهینه حاصل را مشخص کنید.

$N=3,\ M=6,\ (w_1,w_2,w_3)=(2,3,4),\ (p_1,p_2,p_3)=(1,2,5)$

راه حل: باید $g_0(6)$ را حساب کنیم.

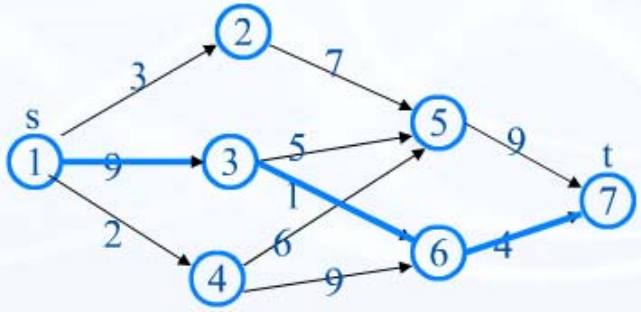
مثال



$O(2^n)$

گراف‌های چند مرحله‌ای

* گراف چند مرحله‌ای گرافی است جهت دار که در آن رئوس V به $k>1$ زیر مجموعه مجزا افراز شده(V_i) و برای هر یال (u,v)، $u\varepsilon V_i$ و $v\varepsilon V_{i+1}$، مجموعه‌های V_1 و V_k هر کدام دارای یک عضو هستند.



گرافهای چند مرحله‌ای (ادامه)

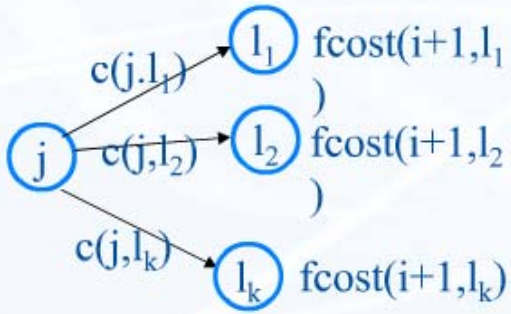
* هر مسیر از s به t را می‌توان به صورت نتیجه یک رشته از k-2 تصمیم در نظر گرفت.
* تصمیم iام مستلزم آن است که مشخص کنیم کدام رأس از مجموعه V_{i+1} باید در مسیر باشد. (1<=i<=k-2)
* فرض کنید که p(i,j) مسیر با هزینه مینیمم از رأس j در V_i به راس t باشد و fcost(i,j) هزینه این مسیر است.

گرافهای چند مرحله‌ای (ادامه)

$$fcost(i,j)= \min_{l \varepsilon V_{i+1},\ (j,l) \varepsilon E} \{c(j,l)+fcost(i+1,l)\}$$

* جواب مسئله به صورت $fcost(1,s)$ است.

* همچنین $fcost(k-1,j)=c(j,t)$



مثال

fcost(1,1)=Min{3+fcost(2,2), 9+fcost(2,3), 2+fcost(2,4)} ={19,14,15}=15

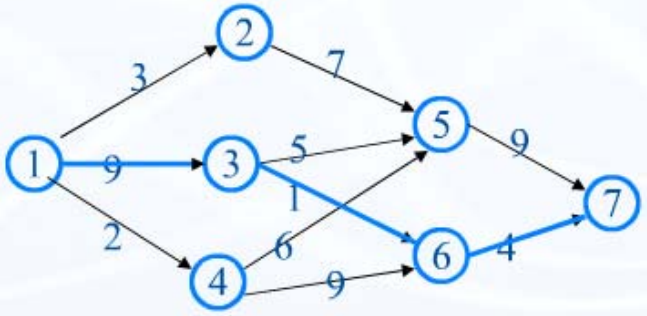
fcost(2,2)=Min{7+fcost(3,5)} ={7+9}=16

fcost(2,3)=Min{5+fcost(3,5), 1+fcost(3,6)} ={5+9, 1+4}=5

fcost(2,4)=Min{6+fcost(3,5), 9+fcost(3,6)} ={6+9, 9+4}=13

fcost(3,5)=9

fcost(3,6)=4



گرافهای چند مرحله‌ای (الگوریتم)

```
Algorithm  FGraph(E,k,n,p)
fcost[n]=0;
For(j=n-1;j>0;j--){
    r | (j,r) ε E & {c[j,r]+fcost[r] is minimum}
    fcost[j]=c[j,r]+fcost[r]
    D[j]=r
}
P[1]=1; p[k]=n;
for(j=2;j<k;j++)
    p[j]=D[p[j-1]]
```

$O(|V| + |E|)$

$O(k)$

$\Rightarrow O(|V| + |E|)$

مساله تمامی کوتاه ترین مسیرها

* فرض کنید G یک گراف جهت دار و وزن دار با n رأس است که در آن $V=\{1,2,\ldots,n\}$ است و c ماتریس هزینه‌های آن است که به صورت زیر تعریف شده است:

$c[i,i]=0$ و $c[i,j]$ برابر وزن یال بین رأس i و j است. (اگر یالی وجود داشته باشد وگرنه مقدار بینهایت)

* هدف بدست آوردن کوتاهترین مسیر بین هر دو رأس دلخواه است.

تمامی کوتاه ترین مسیرها (ادامه)

* مساله تمامی کوتاهترین مسیرها، تعیین یک ماتریس $n \times n$ مانند A است به گونه‌ای که $A[i,j]$ حاوی طول کوتاهترین مسیر از رأس i به راس j است.
* ماتریس A را می‌توان با حل n مسئله کوتاه ترین مسیرهای هم‌مبدا (الگوریتم دایکسترا) بدست آورد.
* با توجه به اینکه الگوریتم دایکسترا از $O(n^2)$ می‌باشد، تکرار آن نیاز به $O(n^3)$ دارد.

الگوریتم فلوید-وارشال

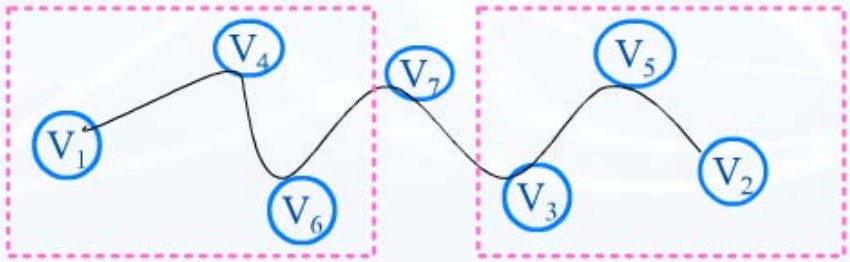
* با استفاده از اصل بهینه سازی می‌توان الگوریتم دیگری با روش برنامه‌سازی پویا برای این مسئله ارائه داد (با $O(n^3)$)

* بر خلاف الگوریتم دایکسترا که همه یالها باید دارای وزن مثبت باشند، در این الگوریتم یالها می‌توانند با وزن منفی باشند اما گراف دارای دور با وزن منفی نباید باشد.

* **اصل بهینه سازی**: اگر k یک رأس میانی در کوتاهترین مسیر از رأس i به رأس j باشد، آنگاه دو زیر مسیر i تا k و k تا j باید کوتاهترین مسیرهای i به k و k به j باشند.

الگوریتم فلوید-وارشال (ادامه)

* **ایده:** اگر k، یک رأس میانی با بزرگترین اندیس باشد، آنگاه مسیر از i به k کوتاهترین مسیری است که از i شروع و به k ختم شده و از هیچ رأس میانی با اندیس بزرگتر از k-1 عبور نمی‌کند. به طور مشابه، مسیر از k به j، کوتاهترین مسیر از k به j است که از هیچ رأس میانی با اندیس بزرگتر از k-1 عبور نمی‌کند.



الگوریتم فلوید-وارشال (ادامه)

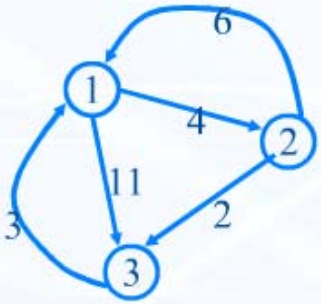
* اما ممکن است چند مسیر متفاوت بین i و j باشد که در یکی از آنها بزرگترین اندیس وجود دارد، در این حال باید بین این دو مسیر مینیمم را انتخاب کرد، یعنی اگر $A_k[i,j]$ طول کوتاهترین مسیر از i به j باشد که در آن k بزرگترین اندیس میانی است، پس:

$$A^k[i,j]=Min\{A^{k-1}[i,k]+ A^{k-1}[k,j], A^{k-1}[i,j]\}$$

$$A[i,j]=Min\{Min\{A^{k-1}[i,k]+ A^{k-1}[k,j], A^{k-1}[i,j]\}\ 1<=k<=n\}$$

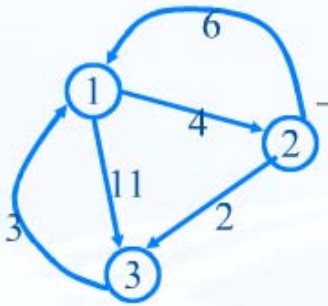
الگوریتم فلوید-وارشال (ادامه)

* بدیهی است که $A[i,j]=A^n[i,j]$ و $A^0[i,j]=C[i,j]$
* به ترتیب با محاسبه A^2 ، A^1و ... و A^n می‌توان پاسخ را بدست آورد.



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

مثال



A^0	1	2	3
1	0	4	11
2	6	0	2
3	3	∞	0

A^1	1	2	3
1	0	4	11
2	6	0	2
3	3	7	0

A^2	1	2	3
1	0	4	6
2	6	0	2
3	3	7	0

A^3	1	2	3
1	0	4	6
2	5	0	2
3	3	7	0

الگوریتم فلوید-وارشال (ادامه)

```
Algorithm  All_Paths(Cost,A,n)
for i=1 to n do
{  for j=1 to n do
      A[i,j]=Cost[i,j];
}
for k=1 to n do
     for i=1 to n do
          for j=1 to n do
               A[i,j]=Min{A[i,j], A[i,k]+A[k,j]}
```

$O(n^3)$

تمرین

برای گراف مقابل مراحل اجرای الگوریتم فلوید را نشان دهید.