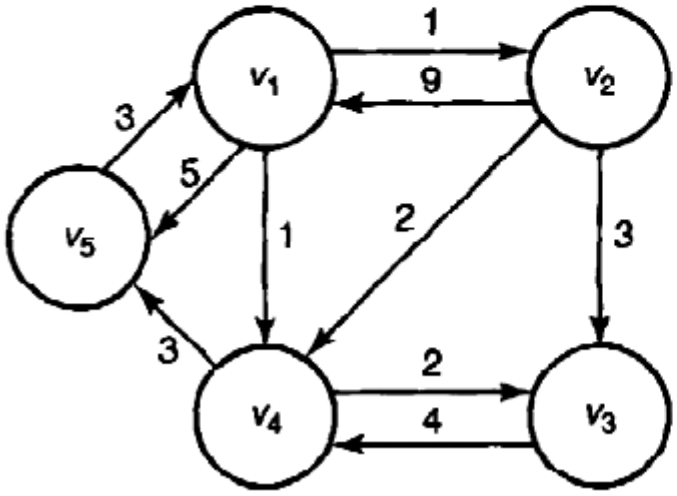
الگوریتم فلوید برای محاسبه کوتاه ترین مسیر ها

- اصطلاحات:
  - گراف : رئوس، یال ها (کمان ها)
  - گراف جهت دار (دایگراف)
  - گراف وزن دار



اصطلاحات

- مسیر
- دور
- گراف دور دار/ بدون دور
- مسیر ساده
- طول یک مسیر

مساله کوتاه ترین مسیر

- **مساله** بهینه سازی
  - می تواند بیش از یک راه حل کاندیدا وجود داشته باشد
  - هر راه حل کاندیدا دارای یک مقدار می باشد.
  - راه حل، یک راه حل کاندیدا می باشد که دارای مقدار بهینه می باشد.

- الگوریتم brute force (الگوریتمی که تمام حالت های ممکن را در نظر می گیرد) دارای پیچیدگی زمانی به صورت فاکتوریل می باشد:

$$(n-2)(n-3)\ldots 1 = (n-2)!$$

نمایش گراف

- ماتریس مجاورتی

$$W[i][j] = \begin{cases} \text{وزن یال} & \text{اگر یالی بین } V_i \text{ و } V_j \text{ باشد} \\ \infty & \text{اگر یالی بین } V_i \text{ و } V_j \text{ نباشد} \\ 0 & \text{اگر } i = j \text{ باشد} \end{cases}$$

	1	2	3	4	5
1	0	1	∞	1	5
2	9	0	3	2	∞
3	∞	∞	0	4	∞
4	∞	∞	2	0	3
5	3	∞	∞	∞	0

W

	1	2	3	4	5
1	0	1	3	1	4
2	8	0	3	2	5
3	10	11	0	4	7
4	6	7	2	0	3
5	3	4	6	4	0

D

الگوریتم

- ایجاد دنباله ای از $n+1$ ماتریس $D^{(k)}$ به طوری که $0 \le k \le n$

$$D^{(0)}, D^{(1)}, D^{(2)}, \ldots, D^{(n)}$$

$D^{(k)}[i][j]$ = طول کوتاه ترین مسیر از V_i به V_j که فقط از رئوس موجود در مجموعه $\{V_1, V_2, \ldots, V_k\}$ به عنوان رئوس میانی استفاده می کند.

$D^{(0)} = W$

$D^{(n)} = D$

مثال

- محاسبه $D[2][5]$

$D^0[2][5] = length[v_2, v_5] = \infty$

$D^1[2][5] = \text{minimum}(length[v_2, v_5], length[v_2, v_1, v_5]$
$= \text{minimum}(\infty, 14) = 14$

$D^2[2][5] = D^1[2][5] = 14$

$D^3[2][5] = D^2[2][5] = 14$

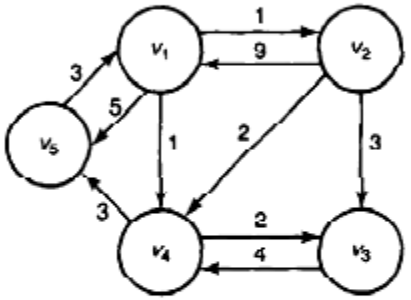


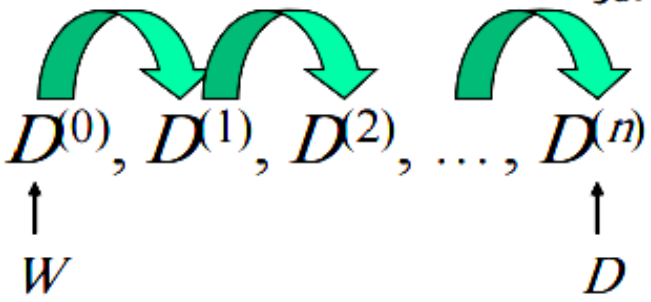
Figure 3.2 ● A weighted, directed graph.

$D^4[2][5] = \text{minimum}(length[v_2, v_1, v_5], length[v_2, v_4, v_5],$
$length[v_2, v_1, v_4, v_5], length[v_2, v_3, v_4, v_5])$
$= \text{minimum}(14, 5, 13, 10) = 5$

$D^5[2][5] = D^4[2][5] = 5$

برنامه نویسی پویا برای مساله کوتاه ترین مسیر

- باید راهی برای محاسبه $D = D^{(n)}$ از روی $W = D^{(0)}$ به دست آوریم:
  - ارایه یک خاصیت بازگشتی برای محاسبه کردن $D^{(k)}$ از $D^{(k-1)}$
  - حل نمونه ای از مساله به روش پایین به بالا بازاء $k = 1$ تا n و ایجاد دنباله زیر:



الگوریتم فلوید

$D^{(0)} = W;$

for ($k = 1$; $k <= n$, k++)

compute $D^{(k)}$ from $D^{(k-1)}$;

محاسبه $D^{(k)}$ از روی $D^{(k-1)}$

- **حالت ۱:** حداقل یک نمونه از کوتاه ترین مسیرها از V_i به V_j که فقط از رئوس موجود در مجموعه $\{V_1, V_2, \ldots, V_k\}$ به عنوان رئوس میانی استفاده می کند، از V_k عبور نکند. در این صورت:

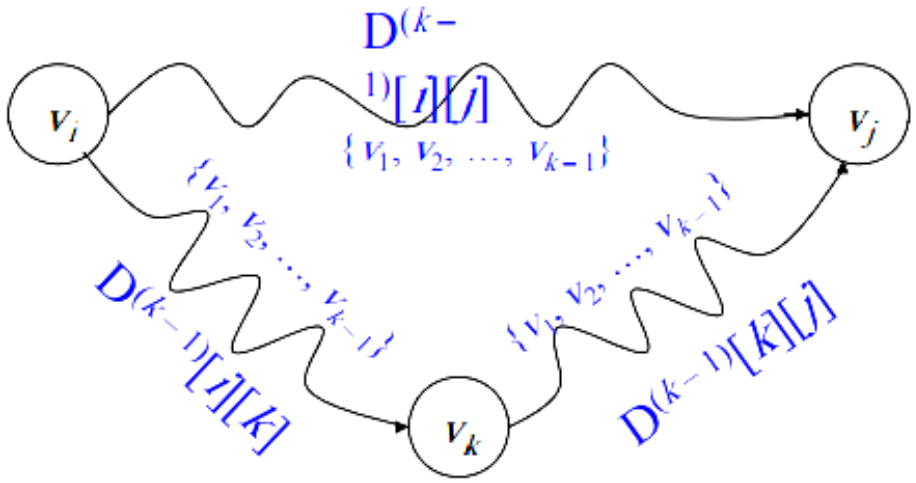
$$D^{(k)}[i][j] = D^{(k-1)}[i][j]$$

- **حالت ۲:** تمام نمونه های کوتاه ترین مسیر از V_i به V_j که فقط از رئوس موجود در مجموعه $\{V_1, V_2, \ldots, V_k\}$ به عنوان رئوس میانی استفاده می کند، از V_k عبور کنند. در این صورت:

$$D^{(k)}[i][j] = D^{(k-1)}[i][k] + D^{(k-1)}[k][j]$$

ارائه خاصیت بازگشتی: محاسبه $D^{(k)}$ از روی $D^{(k-1)}$

$$D^{(k-1)} \Rightarrow D^{(k)}$$



$$D^{(k)}[i][j] = minimum(D^{(k-1)}[i][j], D^{(k-1)}[i][k] + D^{(k-1)}[k][j])$$

مثال

- محاسبه $D^{(2)}[5][4]$

$$D^{(1)}[2][4] = \text{minimum}(D^{(0)}[2][4], D^{(0)}[2][1] + D^{(0)}[1][4]) = \text{minimum}(2, 9 + 1) = 2$$

$$D^{(1)}[5][2] = \text{minimum}(D^{(0)}[5][2], D^{(0)}[5][1] + D^{(0)}[1][2]) = \text{minimum}(\infty, 3 + 1) = 4$$

$$D^{(1)}[5][4] = \text{minimum}(D^{(0)}[5][4], D^{(0)}[5][1] + D^{(0)}[1][4]) = \text{minimum}(\infty, 3 + 1) = 4$$

$$D^{(2)}[5][4] = \text{minimum}(D^{(1)}[5][4], D^{(1)}[5][2] + D^{(1)}[2][4]) = \text{minimum}(4, 4 + 2) = 4$$

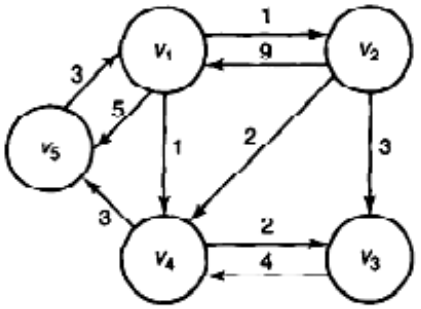


Figure 3.2 • A weighted, directed graph.

محاسبه $D^{(k)}$ از روی $D^{(k-1)}$

```
for (i = 1; i <= n, i++)
    for (j = 1; j <= n, j++)
        D(k)[i][j] = min(D(k-1)[i][j], D(k-1)[i][k] + D(k-1)[k][j]);
```

الگوریتم فلوید

$D^{(0)} = W;$

```
for (k = 1; k <= n; i++)
    for (i = 1; i <= n; i++)
        for (j = 1; j <= n; j++)
            D^(k)[i][j] = min(D^(k-1)[i][j], D^(k-1)[i][k] + D^(k-1)[k][j]);
```

الگوریتم فلوید

$$D^{(k)}[i][j] = minimum(D^{(k-1)}[i][j], D^{(k-1)}[i][k] + D^{(k-1)}[k][j])$$

$$D^{(k)}[k][j] = minimum(D^{(k-1)}[k][j], \overbrace{D^{(k-1)}[k][k]}^{0} + D^{(k-1)}[k][j]) = D^{(k-1)}[k][j]$$

$$D^{(k)}[i][k] = minimum(D^{(k-1)}[i][k], D^{(k-1)}[i][k] + \overbrace{D^{(k-1)}[k][k]}^{0}) = D^{(k-1)}[k][j]$$

پیچیدگی زمانی برای همه حالات

- عمل اصلی: دستور واقع در حلقه for-*j*
- اندازه ورودی: n، تعداد رئوس گراف
- پیچیدگی زمانی:

$$T(n) = n \times n \times n = n^3 \in \Theta(n^3)$$

برنامه نویسی پویا و مسایل بهینه سازی

- مراحل:

– ارائه یک خاصیت بازگشتی برای بدست آوردن راه حل بهینه نمونه ای از مساله

– محاسبه مقدار راه حل بهینه به روش پایین به بالا

– ساختن یک راه حل بهینه به روش پایین به بالا

اصل بهینگی

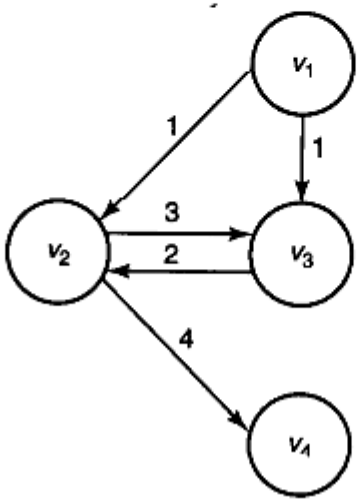
- یک راه حل بهینه برای یک نمونه از مساله همواره شامل راه حل های بهینه برای تمامی زیر نمونه ها می باشد.

- اطمینان می دهد که راه حل بهینه یک نمونه از مساله می تواند با ترکیب راه حل های بهینه زیر نمونه ها حاصل شود.

- قبل از استفاده از برنامه نویسی پویا برای بدست آوردن راه حل، لازم است که نشان دهیم اصل بهینگی برقرار است.

مساله طولانی ترین مسیرها

- طولانی ترین مسیر از V_1 به V_4 مسیر $[V_1, V_3, V_2, V_4]$ می باشد.
- زیر مسیر $[V_1, V_3]$ بهینه نیست.



ضرب زنجیره ای ماتریس ها

- برای ضرب نمودن یک ماتریس $i * j$ در یک ماتریس $j * k$ تعداد ضرب های ابتدایی برابر با $i * j * k$ می باشد.
- ضرب ماتریس ها دارای خاصیت شرکت پذیری می باشد، بدین معنا که ترتیب های متفاوتی برای ضرب چند ماتریس در یکدیگر وجود دارد که هر یک منجر به تعداد متفاوتی از ضرب های ابتدایی می شود.
- مثال:

$$\begin{array}{cccccccc} A & \times & B & \times & C & \times & D \\ 20\times2 & & 2\times30 & & 30\times12 & & 12\times8 \end{array}$$

درخت های جستجوی دودویی بهینه

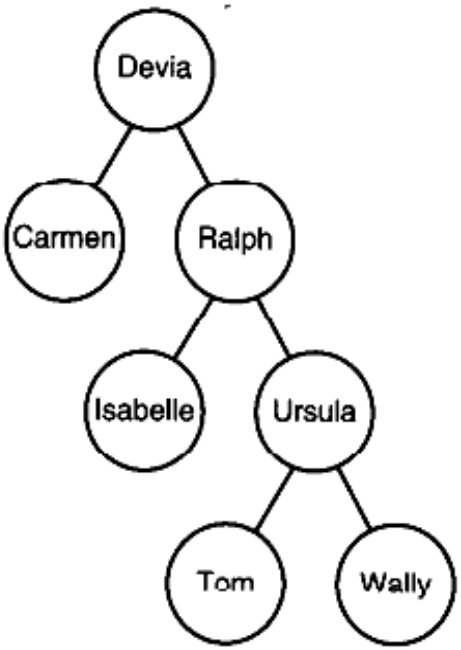
درخت جستجوی دودویی یک درخت دودویی از عناصر (کلید ها) است، که از یک مجموعه مرتب حاصل می شود، به طوری که

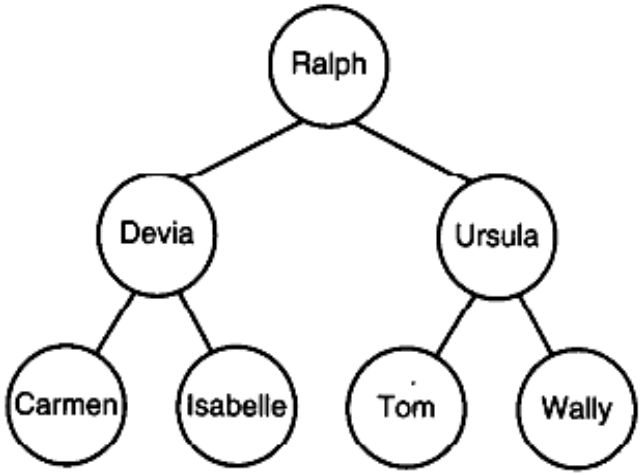
۱. هر گره دارای یک کلید می باشد.

۲. کلیدهای واقع در زیر درخت چپ یک گره، کوچکتر یا مساوی کلید آن گره می باشند.

۳. کلیدهای واقع در زیر درخت راست یک گره، بزرگتر یا مساوی کلید آن گره می باشند.

مثال ها


Devia
Carmen
Ralph
Isabelle
Ursula
Tom
Wally


Ralph
Devia
Ursula
Carmen
Isabelle
Tom
Wally

درخت متوازن

- عمق (سطح) یک گره: تعداد لبه های موجود در مسیر منحصر بفرد از ریشه به آن گره.
- عمق یک درخت: حداکثر عمق تمامی گره ها در آن درخت.
- درخت دودویی متوازن: اگر عمق دو زیر درخت از هر گره بیش از یک واحد اختلاف نداشته باشند.
- درخت جستجوی دودویی بهینه: زمان متوسط برای مکان یابی یک کلید کمینه است.

متوسط زمان جستجو

- زمان جستجو: تعداد مقایسه های انجام شده برای مکان یابی یک کلید
- زمان جستجو برای key برابر است با:

$$depth(key) + 1$$

- متوسط زمان جستجو:

$$\sum_{i=1}^{n} c_i \, p_i$$

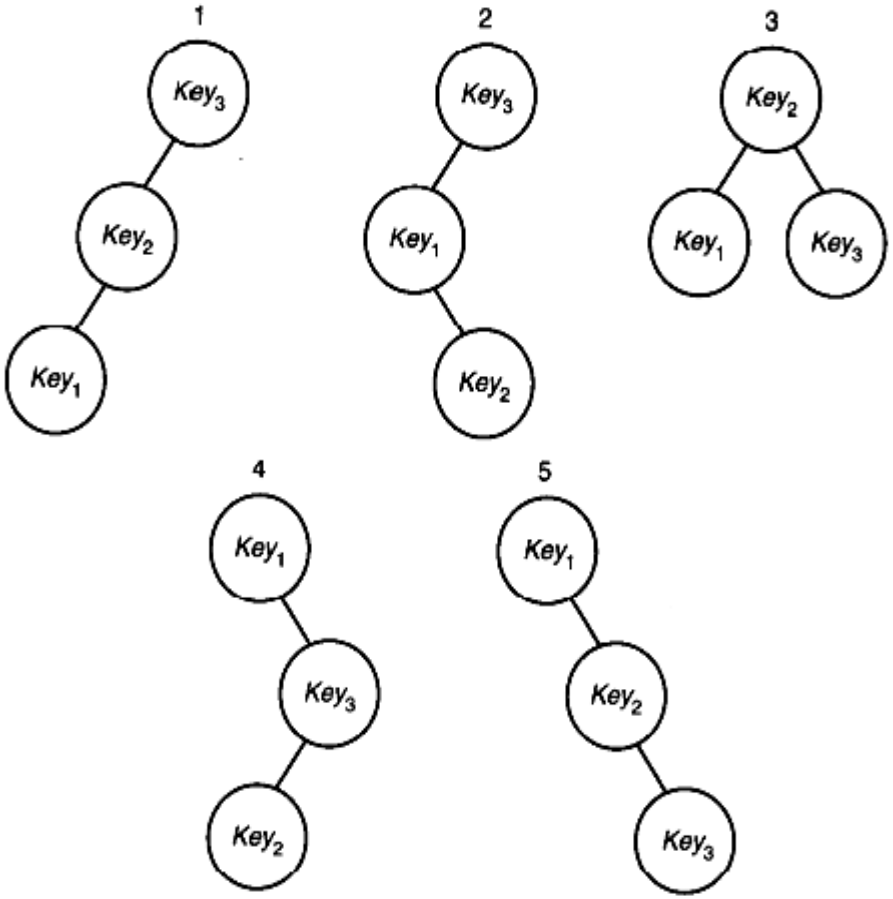
که در آن:

n تعداد کلید ها،

p_i احتمال آنکه key_i کلید مورد جستجو باشد،

c_i تعداد مقایسه های مورد نیاز برای یافتن key_i می باشد.

مثال



- $P_1 = 0.7$
- $P_2 = 0.2$
- $P_3 = 0.1$

1. $3(0.7) + 2(0.2) + 1(0.1) = 2.6$
2. $2(0.7) + 3(0.2) + 1(0.1) = 2.1$
3. $2(0.7) + 1(0.2) + 2(0.1) = 1.8$
4. $1(0.7) + 3(0.2) + 2(0.1) = 1.5$
5. $1(0.7) + 2(0.2) + 3(0.1) = 1.4$

مساله فروشنده دوره گرد (*TSP*)

- تور(دور هامیلتونی): مسیری از یک راس به خودش که از هر یک از رئوس دیگر دقیقا یک بار عبور می کند.
- تور بهینه: مسیری با مشخصات بالا با طول حداقل.
- الگوریتم *Brute force* از مرتبه فاکتوریل می باشد:

$$(n\text{-}1)(n\text{-}2)\cdots 1 = (n\text{-}1)!$$

- اصل بهینگی برقرار است(؟).

یک مثال

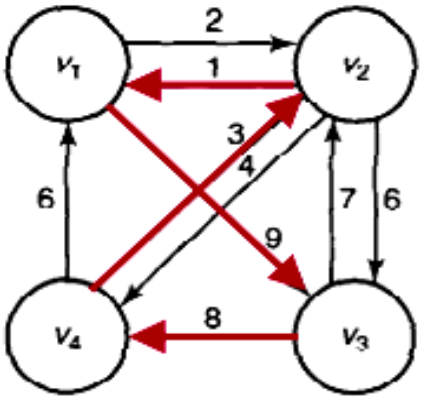


Figure 3.16 ● The optimal tour is $[v_1, v_3, v_4, v_2, v_1]$.

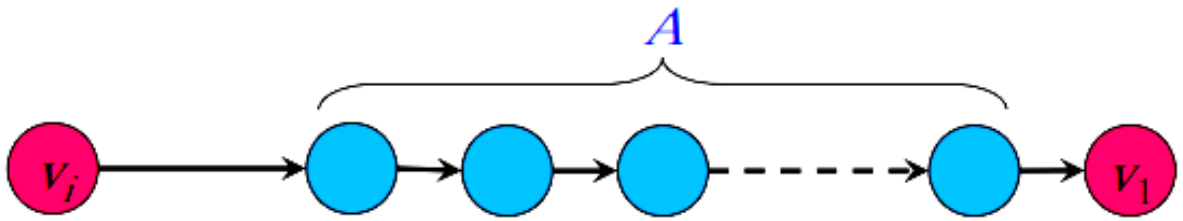
$$length[V_1, V_2, V_3, V_4, V_1] = 22$$

$$length[V_1, V_3, V_2, V_4, V_1] = 26$$

$$length[V_1, V_3, V_4, V_2, V_1] = 21$$

آماده سازی

- V = مجموعه تمام رئوس $V = \{v_1, v_2, \cdots, v_n\}$
- A = زیر مجموعه ای از V
- $D[v_i][A]$ = طول کوتاهترین مسیر از v_i به v_1 که از تمام رئوس مجموعه A دقیقا یک بار عبور می کند.



مثال

- محاسبه $D[v_2][A]$ برای گراف زیر

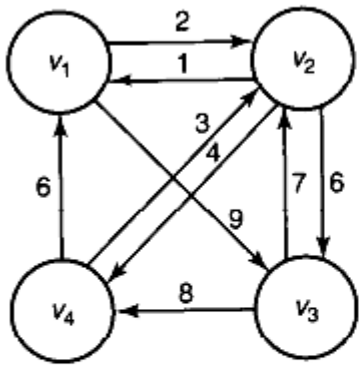


Figure 3.16 ● The optimal tour is $[v_1, v_3, v_4, v_2, v_1]$.

اگر $A=\{v_3\}$، آنگاه:

$$D[v_2][A] = length[v_2, v_3, v_1] = \infty$$

اگر $A=\{v_3, v_4\}$، آنگاه:

$$D[v_2][A] = \text{minimum}(\ length[v_2, v_3, v_4, v_1], length[v_2, v_4, v_3, v_1]\ )$$

$$= \text{minimum}(20, \infty) = 20$$

الگوریتم

- طول یک تور بهینه =

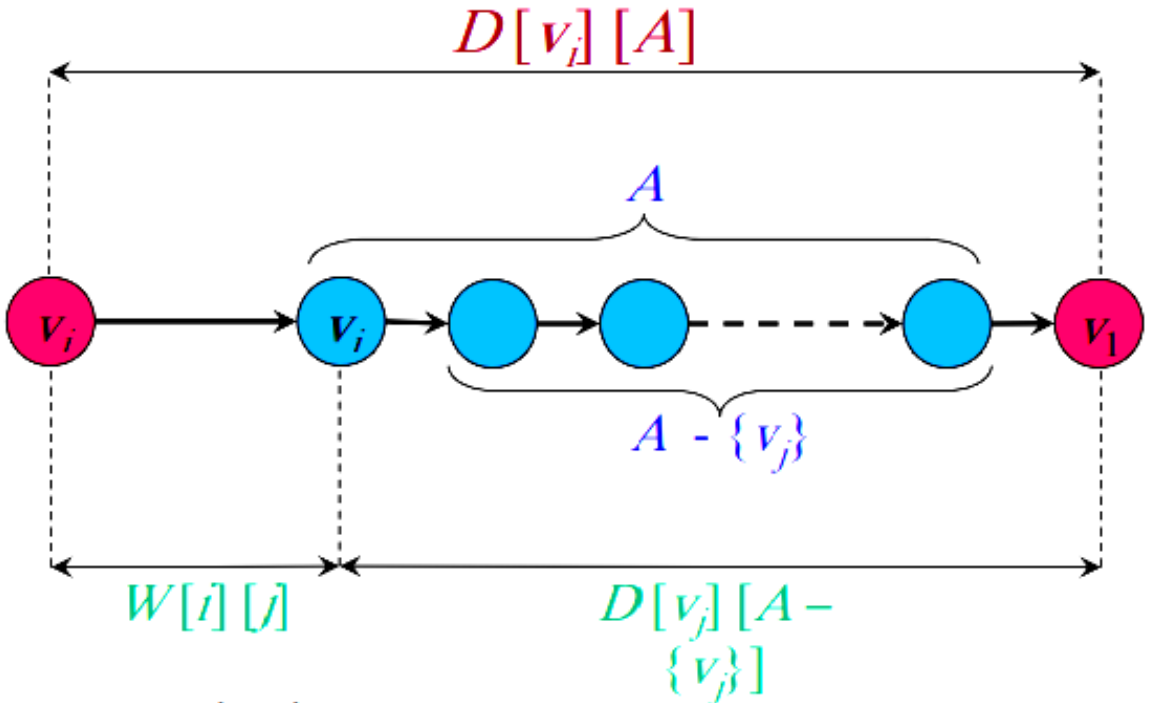
$$D[v_1][V-\{v_1\}] = \underset{2\le j\le n}{\text{minimum}}(W[1][j] + D[v_j][V-\{v_1, v_j\}])$$

- به طور کلی به ازای $i \neq 1$ و $v_i \notin A$

$$D[v_i][A] = \min_{j\cdot v_j \in A} imum(W[i][j] + D[v_j][A-\{v_j\}]) \quad if \quad A \neq \phi$$

$$D[v_i][\phi] = W[i][1]$$

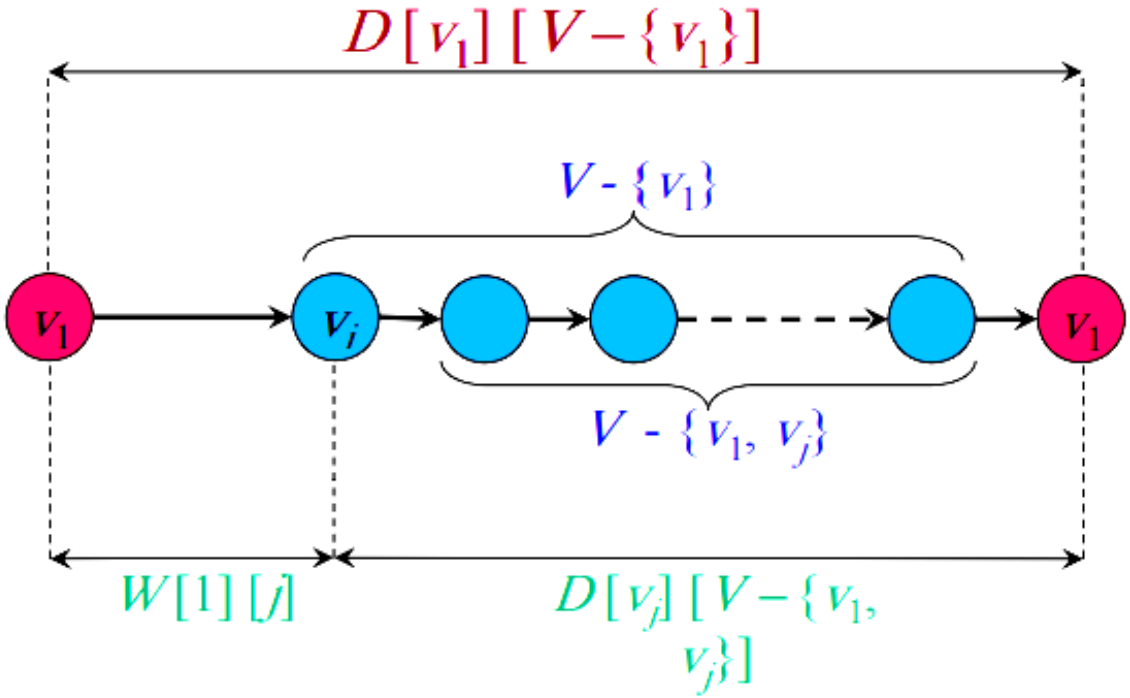
محاسبه طول مسیر بهینه ($i \neq 1$ و $v_j \notin A$)



$$D[v_i][A] = \min_{j:v_j \in A} imum(W[i][j] + D[v_j][A - \{v_j\}]) \quad if \quad A \neq \phi$$

$$D[v_i][\phi] = W[i][1]$$

محاسبه طول تور بهینه



$$D[v_1][V-\{v_1\}] = \underset{2\le j\le n}{\text{minimum}}(W[1][j] + D[v_j][V-\{v_1, v_j\}])$$

مثال

- محاسبه تور بهینه برای گراف زیر

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

مثال فروشنده دوره گرد

	∅	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-			-	-		-
3	∞		-		-		-	-
4	6			-		-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_2][\phi] = 1$$

$$D[v_3][\phi] = \infty$$

$$D[v_4][\phi] = 6$$

مثال فروشنده دوره گرد

	$\emptyset$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-			-	-		-
3	∞	8	-		-		-	-
4	6	4		-		-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_3][\{v_2\}] = \underset{j:v_j \in \{v_2\}}{\text{minimum}}(W[3][j] + D[v_j][\{v_2\} - \{v_j\}])$$

$$= W[3][2] + D[v_2][\phi] = 7 + 1 = 8$$

$$D[v_4][\{v_2\}] = \underset{j:v_j \in \{v_2\}}{\text{minimum}}(W[4][j] + D[v_j][\{v_2\} - \{v_j\}])$$

$$= W[4][2] + D[v_2][\phi] = 3 + 1 = 4$$

مثال فروشنده دوره گرد

	$\varnothing$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-	∞		-	-		-
3	∞	8	-		-		-	-
4	6	4	∞	-		-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_2][\{v_3\}] = \underset{j, v_j \in \{v_3\}}{\text{minimum}}(W[2][j] + D[v_j][\{v_3\} - \{v_j\}])$$

$$= W[2][3] + D[v_3][\phi] = 6 + \infty = \infty$$

$$D[v_4][\{v_3\}] = \underset{j, v_j \in \{v_3\}}{\text{minimum}}(W[4][j] + D[v_j][\{v_3\} - \{v_j\}])$$

$$= W[4][3] + D[v_3][\phi] = \infty + \infty = \infty$$

مثال فروشنده دوره گرد

	$\varnothing$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-	∞	10	-	-		-
3	∞	8	-	14	-		-	-
4	6	4	∞	-		-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_2][\{v_4\}] = \underset{j, v_j \in \{v_4\}}{\text{minimum}}(W[2][j] + D[v_j][\{v_4\} - \{v_j\}])$$
$$= W[2][4] + D[v_4][\phi] = 4 + 6 = 10$$

$$D[v_3][\{v_4\}] = \underset{j, v_j \in \{v_4\}}{\text{minimum}}(W[3][j] + D[v_j][\{v_4\} - \{v_j\}])$$
$$= W[3][4] + D[v_4][\phi] = 8 + 6 = 14$$

مثال فروشنده دوره گرد

	∅	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-	∞	10	-	-		-
3	∞	8	-	14	-		-	-
4	6	4	∞	-	∞	-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_4][\{v_2, v_3\}] = \underset{j: v_j \in \{v_2, v_3\}}{\text{minimum}}(W[4][j] + D[v_j][\{v_2, v_3\} - \{v_j\}])$$

$$= \text{minimum}(W[4][2] + D[v_2][\{v_3\}], W[4][3] + D[v_3][\{v_2\}])$$

$$= \text{minimum}(3 + \infty, \infty + 8) = \infty$$

مثال فروشنده دوره گرد

	$\varnothing$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-	∞	10	-	-		-
3	∞	8	-	14	-	12 4	-	-
4	6	4	∞	-	∞	-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_3][\{v_2, v_4\}] = \underset{j: v_j \in \{v_2, v_4\}}{\text{minimum}}(W[3][j] + D[v_j][\{v_2, v_4\} - \{v_j\}])$$

$$= \text{minimum}(W[3][2] + D[v_2][\{v_4\}], \; W[3][4] + D[v_4][\{v_2\}])$$

$$= \text{minimum}(7 + 10, 8 + 4) = 12$$

مثال فروشنده دوره گرد

	$\varnothing$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	
2	1	-	∞	10	-	-	20 3	-
3	∞	8	-	14	-	12 4	-	-
4	6	4	∞	-	∞	-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_2][\{v_3, v_4\}] = \underset{j: v_j \in \{v_3, v_4\}}{\text{minimum}}(W[2][j] + D[v_j][\{v_3, v_4\} - \{v_j\}])$$

$$= \text{minimum}(W[2][3] + D[v_3][\{v_4\}], W[2][4] + D[v_4][\{v_3\}])$$

$$= \text{minimum}(6 + 14, 4 + \infty) = 20$$

مثال فروشنده دوره گرد

	$\varnothing$	$\{v_2\}$	$\{v_3\}$	$\{v_4\}$	$\{v_2, v_3\}$	$\{v_2, v_4\}$	$\{v_3, v_4\}$	$\{v_2, v_3, v_4\}$
1	-	-	-	-	-	-	-	21 (3)
2	1	-	∞	10	-	-	20 (3)	-
3	∞	8	-	14	-	12 (4)	-	-
4	6	4	∞	-	∞	-	-	-

	1	2	3	4
1	0	2	9	∞
2	1	0	6	4
3	∞	7	0	8
4	6	3	∞	0

$$D[v_1][\{v_2, v_3, v_4\}] = \underset{j\, v_j \in \{v_2, v_3, v_4\}}{\text{minimum}}(W[1][j] + D[v_j][\{v_2, v_3, v_4\} - \{v_j\}])$$

$$= \text{minimum}(W[1][2] + D[v_2][\{v_3, v_4\}],$$
$$W[1][3] + D[v_3][\{v_2, v_4\}],$$
$$W[1][4] + D[v_4][\{v_2, v_3\}])$$
$$= \text{minimum}(2+20, 9+12, \infty+\infty) = 21$$

برنامه نویسی پویا

```
void travel(int n,
            const number W[][],
            index P[][],
            number& minlength)
{
    index i, j, k;
    number D[1..n][subset of V – {v1}];
    for (i = 2; i <= n, i++)
      D[i][∅] = W[i][1];
    for (k = 1; k <= n – 2; k++)
      for (all subsets A ⊆ V - {v1} containing k vertices)
        for (i such that i ≠ 1 and vi is not in A){
          D[i][A] = minimum( W[i][j] + D[j][A – {vj}]);
                    j : vj ∈ A
          P[i][A] = value of j that gave the minimum;
        }
    D[1][V – {v1}] = minimum ( W[1][j] + D[j][V – {v1, vj}]);
                     2 ≤ j ≤ n
    P[1][V – {v1}] = value of j that gave the minimum;
    minlength = D[1][V – {v1}];
}
```

- بازاء هر $n \geq 1$

$$\sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$$

$$k\binom{n}{k} = n\binom{n-1}{k-1}$$

$$\sum_{k=1}^{n} k\binom{n}{k} = \sum_{k=1}^{n} n\binom{n-1}{k-1} = n\sum_{k=0}^{n-1}\binom{n-1}{k} = n2^{n-1}$$

پیچیدگی زمانی برای همه حالات

- عمل اصلی: دستورالعمل اجرا شده برای هر مقدار از V_j
- اندازه ورودی: n، تعداد رئوس در گراف
- پیچیدگی زمانی:

$$T(n) = \sum_{k=1}^{n-2} (n-1-k)k\binom{n-1}{k}$$

$$(n-1-k)\binom{n-1}{k} = (n-1)\binom{n-2}{k}$$

$$\Rightarrow T(n) = (n-1)\sum_{k=1}^{n-2} k\binom{n-2}{k}$$

$$T(n) = (n\text{-}1)(n\text{-}2)2^{n\text{-}3} \in \Theta(n^2 2^n)$$

- پیچیدگی حافظه:

$$M(n) = 2 \times n2^{n\text{-}1} = n2^n \in \Theta(n2^n)$$

آیا چنین الگوریتمی می تواند مفید باشد؟

- رالف و نانسی در حال رقابت برای تصاحب یک موقعیت شغلی
- مسابقه برای تصاحب موقعیت شغلی در شرکت:
  - هرکسی که سریعتر سفر به ۲۰ شهر مشخص را انجام دهد و به شرکت برگردد برنده است. بین هر دو شهر یک جاده وجود دارد.
  - رالف : الگوریتم غیر هوشمند

$(20-1)!\ \mu s = 19!\ \mu s = 3857$ سال

  - نانسی: الگوریتم برنامه نویسی پویا

$(20-1)(20-2)2^{20-3}\ \mu s = 45$ ثانیه

$20 * 2^{20} = 20,971,520$ عنصر آرایه

اگر تعداد شهر ها 60 باشد: الگوریتم برنامه نویسی پویا نیز سال ها وقت لازم دارد.