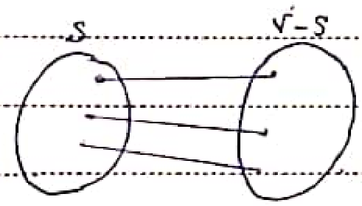
می‌گیرد، پس بجواب $d[i] = w_{xi} + d[x]$ کم‌ترین مقداری است که می‌تواند داشته باشد ⇐ حکم I اثبات شد ✓

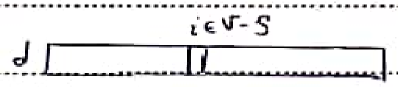
در واقع در دایسترا اصلاً به حکم II نیاز نداشتیم!

جلسه‌ی بیستم: (روز دانشجو مبارک!)

پیاده‌سازی دایسترا

در $O(n^2)$: با یک آرایه d را پیاده‌سازی می‌کنیم:





① $u = \underset{i \in V-S}{\operatorname{argmin}} \{d[i]\}$ — $O(n)$

② $S = S + \{i\}$, update(d) — $O(n)$

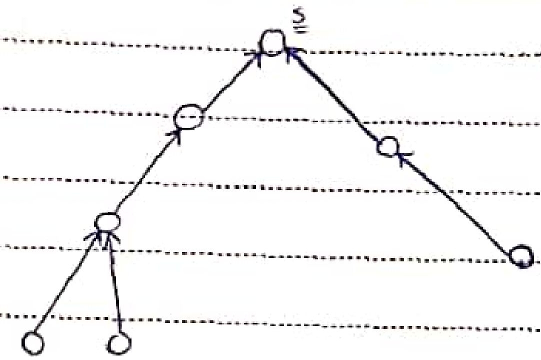
ما باید n بار این کارها را انجام دهیم

```
for i = 1 to n
    d[i] = ∞
d[s] = 0
S = Φ
while |S| ≠ |V| {
    u = argmin {d[i]}   → O(n)
        i∈V-S
    → if d[u] == ∞
        break;
    S = S + {u}
    for i ∈ V - S                 ⎫
        if d[i] > d[u] + wui      ⎬ O(n)
            d[i] = d[u] + wui     ⎪
            parent[i] = u         ⎭
```

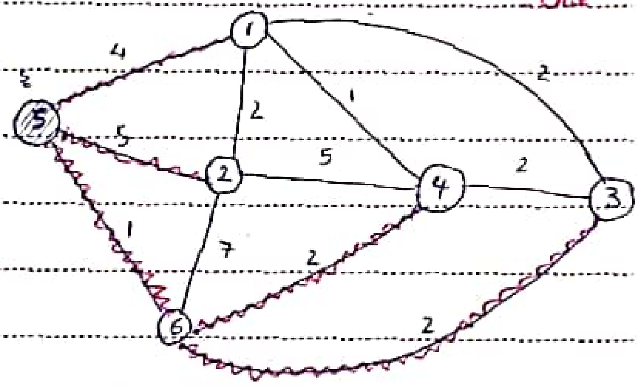
Subject:
Year. Month. Date. ( )

درخت T را فرض کنید. در درخت T همه رأس‌ها با Shortest Path به s متصل شده‌اند:

به این درخت Shortest Path Tree می‌گویند



مثال:

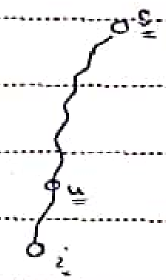


	1	2	3	4	5	6
d	4	5	3	3	0	1

	1	2	3	4	5	6
parent	5	5	6	6	0	5

مقدارهای آرایه d و parent

نکته:



← اگر این مسیر shortest path از s به i باشد، قطعاً shortest path از s به u هم

جست ← همه shortest pathes ها (برای همه گره‌ها) را می‌توان در یک shortest path Tree

دایکسترا:

پیاده‌سازی O(m.log n)

متداول‌تر

2000

Subject:
Year. Month. Date. ( )

d[i] ها را به جای اینکه در یک آرایه نگه داریم، در یک Heap نگه داریم و هر بار آن را آپدیت کنیم

توجه: در بخش update فقط همسایه‌های u را آپدیت می‌کنیم نه همه‌ی گره‌ها را

```
for i=1 to n
    if i ≠ S
        Add((∞, i))
    else
        Add((0, i))
S = ϕ

while |S| ≠ |V| {

    (u, value) = getMin() → O(1)
    delete Min() → O(log n)

    S = S + {u}
    T = T + {(u, parent[u])}

    for all neighbours of u such as i        → • چطور همسایه‌های u را بیابیم؟
        if i.value > u.value + w_ui               با لیست‌ها درست
            i.value = u.value + w_ui              • چطور جای u در Heap را بیابیم
            parent[i] = u                         که بتوانیم آن را update کنیم؟
                                                  با نگه داشتن جای گره در Heap
}                                                 در یک آرایه
```

$O(d_u \log n)$

درجه‌ی رأس u

2000

Subject:
Year Month Date ( )

مزیت دایسترا = نسبتاً سرعتش خوب است + نسبت به الگوریتم های Shortest path دیگر

عیب دایسترا = فرض کرده که یال منفی نداریم + بدون این فرض کار نمی کند

مسئله بعد :

همین Shortest Path ولی این دفعه یال منفی داریم ..

جلسه بیست و یکم

در مسئله Shortest path بدون یال منفی در کتاب از Fib Heap استفاده کرده

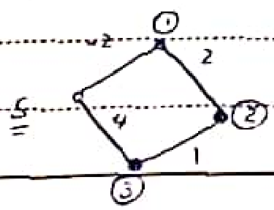
getmin()
updatevalue → Priority Queue → Heap با حل → $m \log n$
Add() → Fib Heap حل کتاب → $m + n \log n$

↓ بهترین پیاده سازی دایسترا

مسئله Shortest Path با وجود یال منفی

$D[i,k]$ = کوتاه ترین مسیر از s به i که دقیقاً k یال دارد = $\min\{ D[j, k-1] + w_{ji} \}$
j چسبیده به i

یال های تکراری هم حساب می شوند ⟹ دور تکراری به وجود می آید ⟹ این الگوریتم



غلط است

Subject:

Year. Month. Date. ( )

0	0	∞	∞	∞
1	∞	-2	∞	4
2	-4			
3				0

از S به 1 برگرد به S و برگرد به 1

دور منفی داره ! X

این الگوریتم اصلاح شده نمیتونه کار کنه : (مثل اینکه اینم جواب نمی ده !)

$D[i,k]$ = کوتاه‌ترین مسیر از s به i که حداکثر k یال داره $= \min\left(\min_{j \in \text{همسایه } i} \{ D[j,k-1] + w_{ji} \} , D[i,k-1] \right)$

با $O(n^3$ یا $nm)$

الگوریتم صحیح جواب :

```
relax(i,j){
    if D[i] + wij < D[j]
        D[j] = D[i] + wij
        Parent[j] = i
}
```

∞	0	∞

$\rightarrow O(nm)$

```
for iter = 1 to n-1
    for all edges such as (i,j)
        relax(i,j)
```

الگوریتم بلمن فورد :

Bellman Ford

(احتمالاً این الگوریتم هم فقط روی گراف جهت دار جواب می ده)

2000

جلسه‌ی بیست و دوم

از این به بعد مسائلی که بررسی می‌کنیم روی گراف‌های جهت‌دار است!

حتی اگر گرافی بدون جهت بود به جای هر یال آن دو یال در دو جهت متفاوت می‌گذاریم

الگوریتم جلسه‌ی پیش:

$$\underbrace{A[u,k]}_{O(n^2)} = \min\left\{ A[u,k] ,\ \overbrace{\min_{v \to u} \{A[v,k-1] + w_{vu}\}}^{O(d_u)} \right\}$$

کوتاه‌ترین مسیر از s به u با حداکثر k یال

$\Downarrow$

پیچیدگی الگوریتم $= O(n) . O(m) = O(n.m)$

اثبات درستی الگوریتم

کوتاه‌ترین

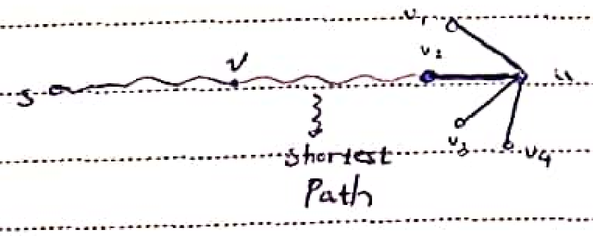
۱. یا مسیر از s به u، k یال دارد

۲. یا " " " " کمتر از k یال دارد

$= A[u,k]$

بین این ۲ داریم min می‌گیریم

رفت و برگشت روی یک یال هم نداریم



2000