ساختمان داده ها

دانشگاه صنعتی نوشیروانی بابل

دکتر حسام عمران پور

طراحان اسلاید:
زهرا ریحانیان و دانیال علیزاده

حل تمرین :
علی باقری

لینک کانال تلگرام اطلاع رسانی و حل تمرین:
t.me/ds_nit_4011

فصل ۱۰

کراف

تعریف گراف

هر گراف G ساختاری شامل دو مجموعه است :

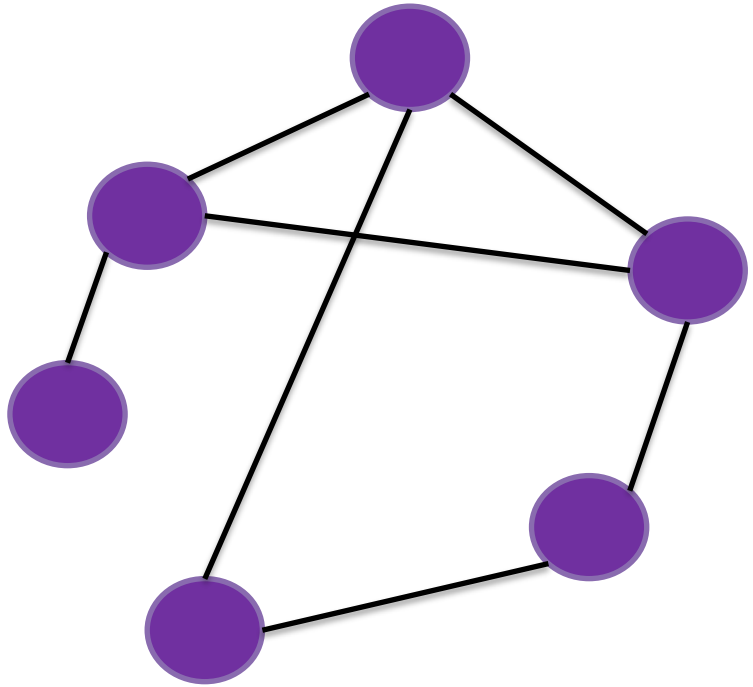
مجوعه راس ها (V)

مجوعه یال ها (E)

$$G = (V , E)$$

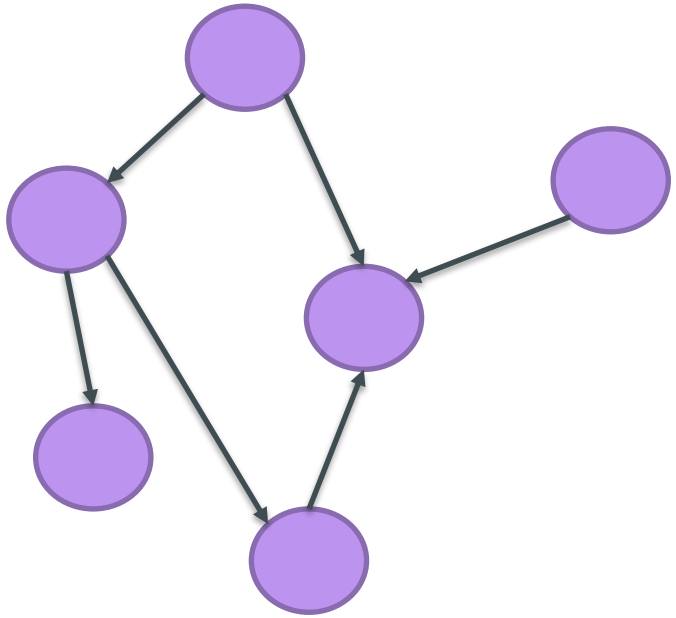
گراف ساده

گرافی که چندگانه و طوقه دار نباشد

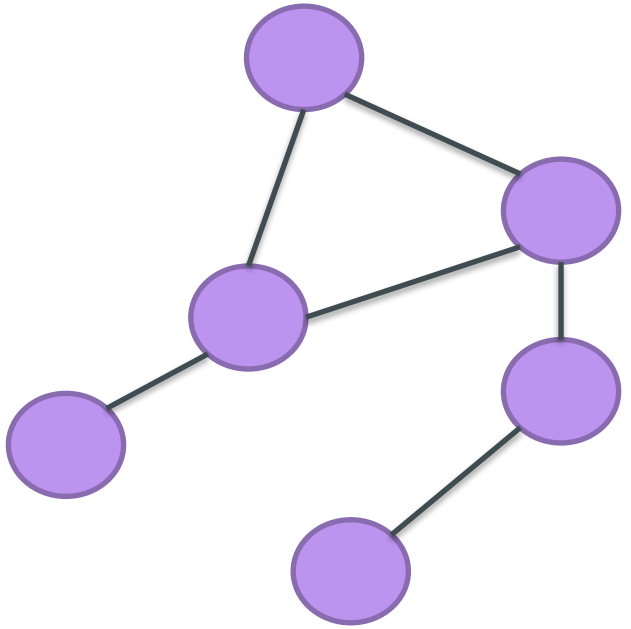


گراف جهت دار و غیر جهت دار

اگر در گراف G ، یال جهت دار وجود داشته باشد ، گراف G را جهت دار
و در غیر این صورت ، آن را گراف غیر جهت دار می نامند.

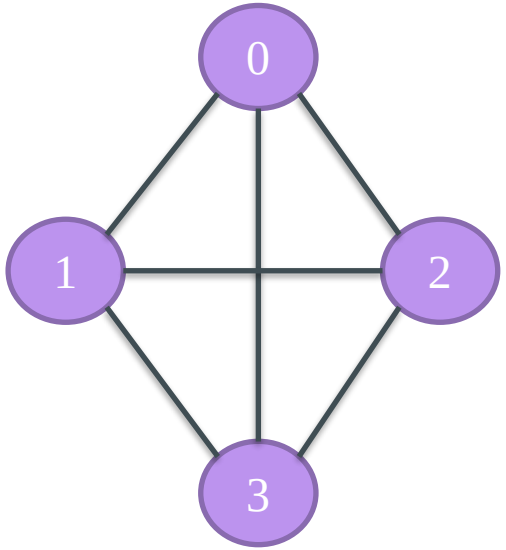


گراف جهت دار



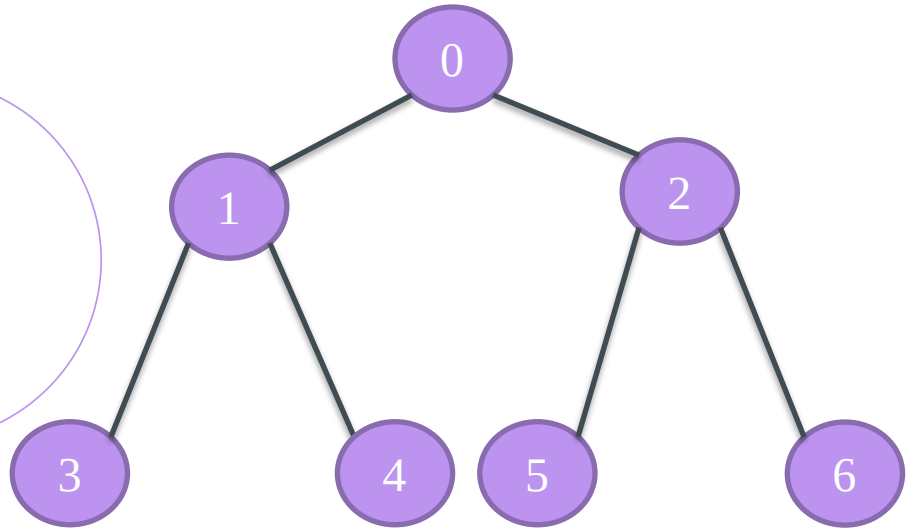
گراف غیر جهت دار

مثال : در شکل های زیر سه گراف G1 ، G2 و G3 نشان داده شده است .



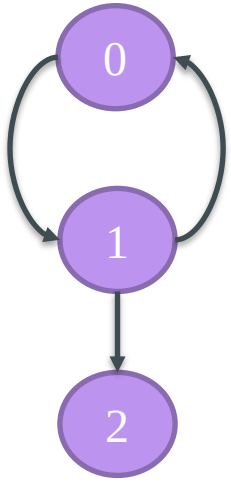
G1

V(G1) = {0, 1, 2, 3}
E(G1) = {(0, 1), (0, 2), (0, 3), (1, 2), (1, 3), (2, 3)}



G2

$V(G2) = \{0, 1, 2, 3, 4, 5, 6\}$
$E(G2) = \{(0, 1), (0, 2), (1, 3), (1, 4), (2, 5), (2, 6)\}$



G3

$V(G3) = \{0, 1, 2\}$
$E(G3) = \{\langle 0, 1\rangle, \langle 1, 0\rangle, \langle 1, 2\rangle\}$

پیاده سازی گراف

پیاده سازی گراف با استفاده از ماتریس مجاورت

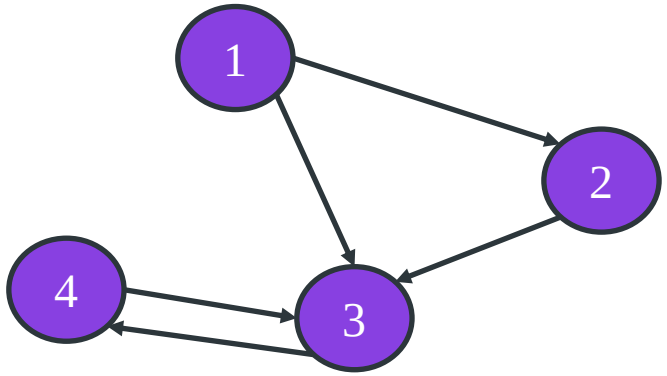
یک ماتریس با اندازه ی |V| * |V| که راس ها به طور دلخواه از ۱ تا |V| شماره گذاری شده باشند. مقدار هر درایه از ماتریس به صورت زیر تعریف می شود :

$$a_{i\,j} = \begin{cases} 1 & (i,j)\,\epsilon\, E, \\ 0 & (i,j)\,\notin\, E, \end{cases}$$

فضای حافظه ی مورد نیاز

$$\boldsymbol{\theta\,(V^2 \times sizeOfDataType)}$$

مثال



	1	2	3	4
1	0	1	1	0
2	0	0	1	0
3	0	0	0	1
4	0	0	1	0

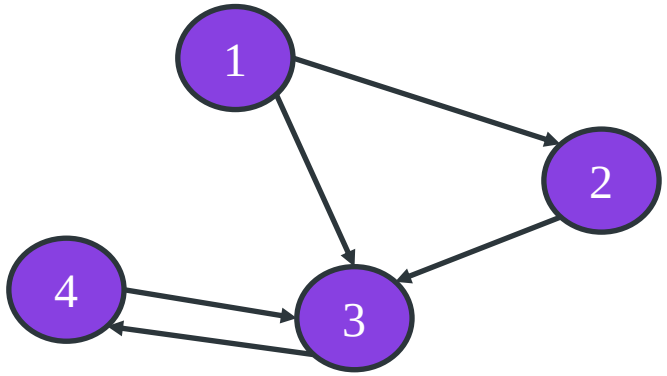
پیاده سازی گراف با استفاده از لیست مجاورت

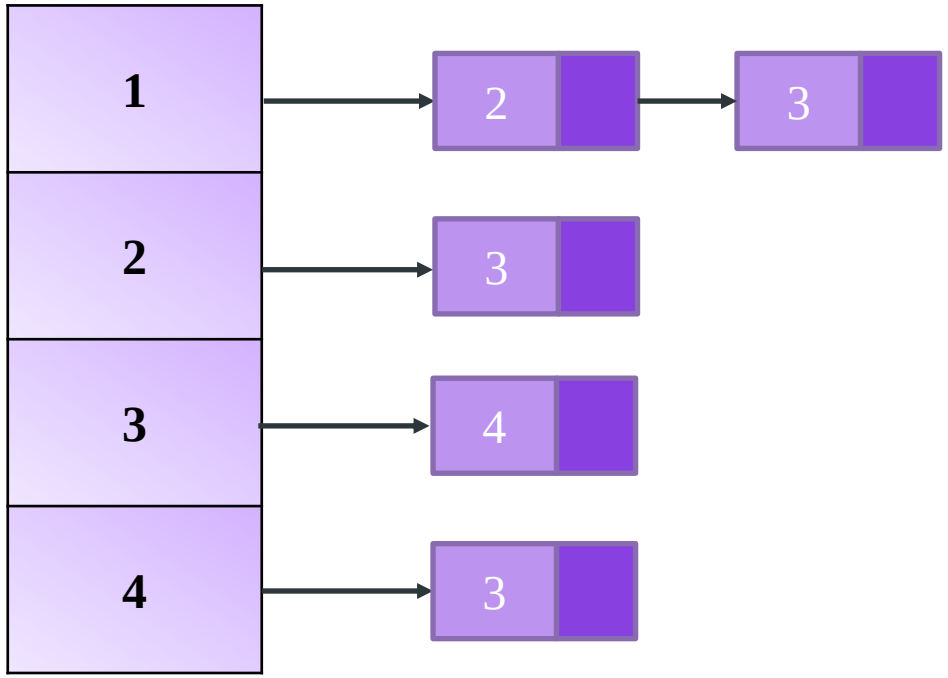
در نمایش گراف G = (V, E) به روش لیست مجاورت، |V| لیست در نظر گرفته می شود. هر یک از این |V| لیست ، یال های یکی از راس ها را نگهداری می کند، به طوری که هر عنصر لیست ، یکی از همسایه های آن راس است. برای پیاده سازی راحت تر، راس ها را به طور دلخواه از 1 تا |V| شماره گذاری کرده و |V| لیست را در یک آرایه به نام Adj قرار می دهند. برای هر راس u ، لیست متناظر آن در Adj[u] قرار دارد. بنابراین هر عنصر لیست Adj[u] ، شماره ی یکی از همسایه های راس u را نگهداری می کند.

فضای حافظه ی مورد نیاز

$$\boldsymbol{\theta\left((V+E)\times sizeOfPointer + V\times sizeOfDataType\right)}$$

مثال





? با توجه به اطلاعات زیر حداکثر تعداد یال ها چقدر باشد تا برای یک گراف با ۱۰ گره ، روش لیست مجاورت حافظه کمتری مصرف کند ؟

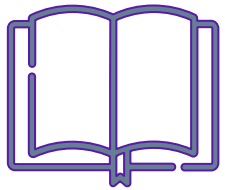
Size of pointer = 5
Size of data = 2

$$\big((10+E)*5\big)+(10*2)<10^2*2$$

$$70+5*E<200$$

$$E<\frac{200-70}{5}$$

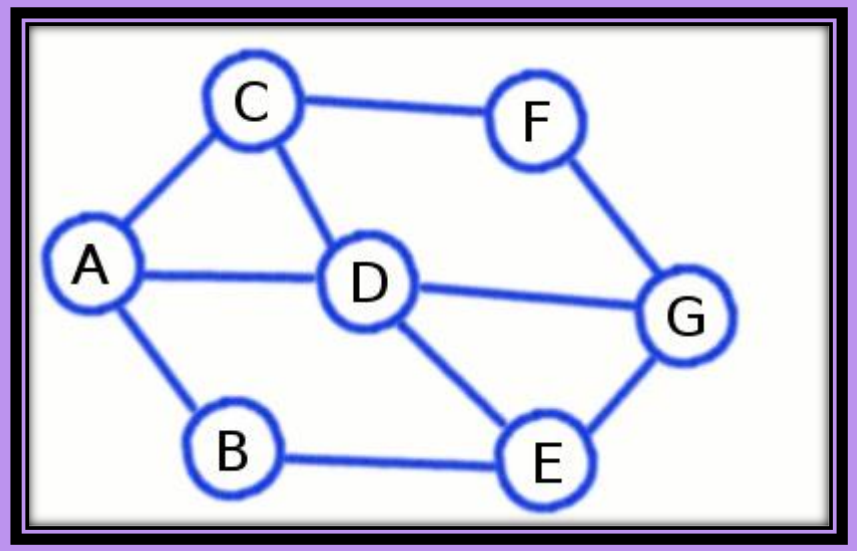
$$E=25$$



روش ماتریس مجاورت برای نمایش گراف های **چگال** (با تعداد یال زیاد) مناسب تر است ، حال آنکه روش لیست مجاورت برای نمایش گراف های **تُنُک** (با تعداد یال کم) کاربرد بیشتری دارد.

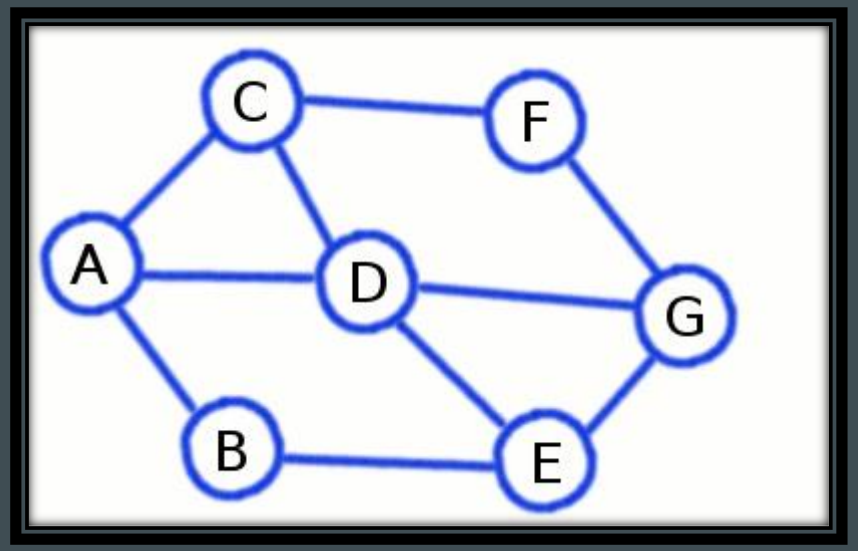
جستجو در گراف

روش های جستجو در گراف



جستجوی سطحی

Breadth-first search (BFS)



جستجوی عمقی

Depth-first search (DFS)

پیاده سازی گراف با استفاده از لیست مجاورت برای استفاده در BFS

```
Const WHITE = 0;
Const GRAY = 1;
Const BLACK = 2;

Type VERTEX = Record
{
    Color : 0 .. 2;
    Distance :integer;
    Parent : integer;
}
Type GRAPH = Record
{
    V : VERTEX[N];
    Adj : LIST[N];
}
```

الگوریتم BFS

```
BFS ( G : GRAPH; source : integer ){
    var u, v : integer;
    var Q : QUEUE;
    var P : Position;
    for i = 1 to |G.V| do {
        G.V[i].Color = WHITE;
        G.V[i].Distance = ∞;
        G.V[i].Parent = NULL;
    }
    G.V[source].Color = GRAY;
    G.V[source].Distance = 0;
    CREATE( Q );
    ENQUEUE( Q, source );
```

```
    while not EMPTY( Q ) do {
        u = DEQUEUE( Q );
        p = FIRST( G.Adj[u] );
        while p ≠ NULL do {
            v = RETRIEVE( G.Adj[u], p );
            if G.V[v].Color == WHITE then{
                G.V[v].Color = GRAY;
                G.V[v].Distance = G.V[u].Distance+1;
                G.V[v].Parent = u;
                ENQUEUE( Q, v );
            }
            p = NEXT( G.Adj[u], p );
        }
        G.V[u].Color = BLACK;
    }
}
```

پیاده سازی گراف با استفاده از لیست مجاورت برای استفاده در DFS

```
Const WHITE = 0;
Const GRAY = 1;
Const BLACK = 2;

Type VERTEX = Record
{
    Color : 0..2;
    Parent : integer;
}
Type GRAPH = Record
{
    V : VERTEX[N];
    Adj : LIST[N];
}
```

الگوریتم DFS

```
DFS ( G : GRAPH )
{
    for u = 1 to |G.V| do
    {
        G.V[u].Color = WHITE;
        G.V[u].Parent = NULL;
    }
    for u = 1 to |G.V| do
    {
        if G.V[u].Color == WHITE then
            DFS-VISIT( G, u );
    }
}
```

```
DFS-VISIT ( G : GRAPH; u : integer )
{
    var p : Position;
    var v : integer;

    G[u].Color = GRAY;
    p = FIRST( G.Adj[u] );
    while p ≠ NULL do
    {
        v = RETRIEVE( G.Adj[u], p );
        if G.V[v].Color == WHITE then  {
            G.V[v].Parent = u;
            DFS-VISIT( G, v );
        }
        p = NEXT( G.Adj[u], p );
    }
    G.V[u].Color = BLACK;
}
```

پایان