Semillero de Programación

Arreglos, Vectores y Grafos

Ana Echavarría Juan Francisco Cardona

Universidad EAFIT

8 de febrero de 2013

Contenido

Arreglos de C++

- Un arreglo es una serie de elementos del mismo tipo ordenados en una secuencia lineal.
- El número de elementos del arreglo es fijo.
- Se puede acceder a cada elemento de manera individual usando el índice de su posición, empezando por el índice 0.
- Por ejemplo, un arreglo de 10 posiciones llamado a puede ser representado así:

Arreglos de C++

Un arreglo debe ser declarado antes de usarse. Para declararse se hace lo siguiente.

Declaración de arreglos

```
tipo_de_dato nombre [número_de_elementos];
```

Ejemplos:

```
int a [10];
string palabras [50];
```

Arreglos de C++

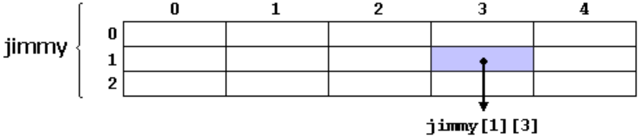
Se pueden declarar arreglos de varias dimensiones

Arreglos de varias dimensiones

```
tipo_de_dato nombre [tam_dim_1] ... [tam_dim_n];
```

Ejemplo:

```
int jimmy [3][5];
```



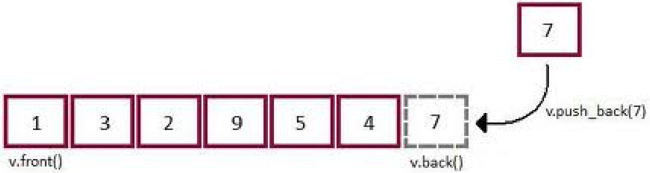
Arreglos de C++

El problema Next Round se puede resolver usando arreglos (http://www.codeforces.com/problemset/problem/158/A)

```
using namespace std;
#include <iostream>

const int MAXN = 55;
int a [MAXN];

int main(){
    int n, k;
    cin >> n >> k;
    for (int i = 0; i < n; i++) cin >> a[i];

    int count = 0;
    int min_score = a[k-1];
    for (int i = 0; i < n; i++){
        if (a[i] >= min_score and a[i] > 0) count++;
        else break;
    }
    cout << count << endl;

return 0;
}
```

Vectores de C++

- Los vectores son contenedores que almacenan los datos en una secuencia pero que pueden cambiar de tamaño.
- Al igual que los arreglos, los elementos pueden ser accedidos por medio del índice de su posición, empezando por en índice 0.
- Contrario a los arreglos, el tamaño de los vectores cambia dinámicamente. Esto hace que utilicen más memoria para poder crecer eficientemente.



Vectores de C++

Para utilizar un vector es necesario incluir la librería vector.

```
#include <vector>
```

Declaración de vector

```
vector <tipo_de_dato> nombre;
```

Ejemplos:

```
vector <int> a;
vector <string> palabras (50);
vector <int> zeros (500, 0);
```

Vectores de C++

El problema Next Round se puede resolver usando vectores (http://www.codeforces.com/problemset/problem/158/A)

```
using namespace std;
#include <iostream>
#include <vector>

vector <int> a;

int main(){
    int n, k;
    cin >> n >> k;
    a.clear();
    for (int i = 0; i < n; i++){
        int ai;
        cin >> ai;
        a.push_back(ai);
    }

    int count = 0;
    int min_score = a[k-1];
    for (int i = 0; i < n; i++){
        if (a[i] >= min_score and a[i] > 0) count++;
        else break;
    }
    cout << count << endl;

return 0;
}
```

Arreglos y vectores en C++

Pregunta

¿Qué significa `vector <int> g [1000]`?

Arreglos y vectores en C++

Pregunta

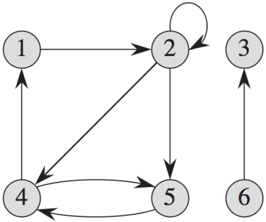
¿Qué significa `vector <int> g [1000]`?

Respuesta

Es un arreglo de 1000 posiciones en el que cada posición contiene un vector de enteros

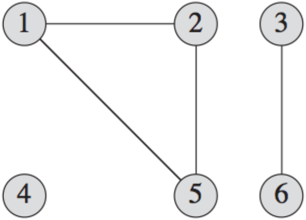
Grafos dirigidos

Un grafo dirigido G es un par (V, E) donde V es un conjunto finito de **nodos** (vertices) y E es un conjunto de **parejas ordenadas** donde cada elemento es un elemento de V y es llamado conjunto de **aristas** (edges).
En la siguiente figura $V = \{1, 2, 3, 4, 5, 6\}$ y
$E = \{(1, 2), (2, 2), (2, 4), (2, 5), (4, 1), (4, 5), (5, 4), (6, 3)\}$



Grafos no dirigidos

En un grafo no dirigido $G = (V, E)$, V es un conjunto finito de **nodos** (vertices) y E es un conjunto de **parejas no ordenadas** (u, v) donde $u, v \in V$ y $u \neq v$.
En la siguiente figura $V = \{1, 2, 3, 4, 5, 6\}$ y
$E = \{(1, 2), (1, 5), (2, 5), (6, 3)\}$



Representación de grafos

Hay dos formas de representar los grafos (tanto dirigidos como no dirigidos) con una matriz de adyacencia o con una lista de ayacencia.
La lista de adyacencia es más comúnmente usada porque usa menos memoria que la matriz de adyacencia.

Matriz de adyacencia

La representación de un grafo $G = (V, E)$ como matriz de adyacencia consiste en una matriz M de tamaño $|V| \cdot |V|$ donde se asume que los nodos están numerados de $0 \ldots |V|$. La matriz de adyacencia se define así:

$$M[u][v] = \begin{cases} 1 & \text{si } (u, v) \in E \text{ o} \\ & \text{si } (v, u) \in E \text{ para grafos no dirigidos} \\ 0 & \text{en caso contrario.} \end{cases} \tag{1}$$

Esto quiere decir que la matriz contiene un 1 si u es adyacente a v y un 0 si no lo es.

Matriz de adyacencia

Pregunta

¿Cuánto espacio requiere la representación de matriz de adyacencia para un grafo $G = (V, E)$?

Matriz de adyacencia

Pregunta

¿Cuánto espacio requiere la representación de matriz de adyacencia para un grafo $G = (V, E)$?

Respuesta

$|V|^2$, se necesita una matriz de tamaño $|V| \cdot |V|$.

Lista de adyacencia

La representación de un grafo $G = (V, E)$ como lista de ayacencia consiste en un arreglo Adj de $|V|$ vectores.

Para cada $u \in V$, $Adj[u]$ contiene una lista (vector) con todos los elementos $v \in V$ tales que $(u, v) \in E$, es decir, todos los nodos adyacentes a u.

En los grafos no dirigidos, dada la arista (u, v) se agregaría v a $Adj[u]$ y u a $Adj[v]$.

Lista de adyacencia

Pregunta

¿Cuánto espacio requiere la representación de lista de adyacencia para un grafo $G = (V, E)$?

Lista de adyacencia

Pregunta

¿Cuánto espacio requiere la representación de lista de adyacencia para un grafo $G = (V, E)$?

Respuesta

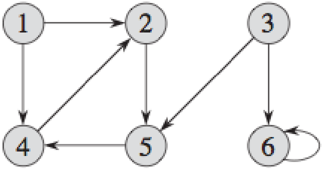
- $|E|$ para grafos dirigidos. Por cada arista se agrega un valor nuevo a la lista de adyacencia.
- $2 \cdot |E|$ para grafos no dirigidos. Por cada arista se agregan dos valores a la lista de adyacencia.

Lista de adyacencia vs. Matriz de adyacencia

- La representación como lista de adyacencia es más común en grafos dispersos, cuando el número de aristas es pequeño ($|E|$ es mucho menor que $|V|^2$).
- La representación como matriz de adyacencia es más común en grafos densos, cuando el número de aritas es grande ($|E|$ es cercano a $|V|^2$).
- Verificar si dos nodos están conectados en un la matriz de adyacencia es más fácil que hacerlo en la lista de adyacencia.

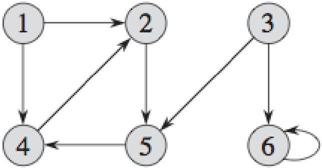
Ejercicio

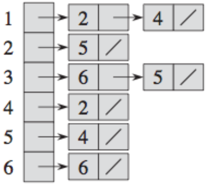
Representar el siguiente grafo dirigido por medio de su matriz de adyacencia y su lista de adyacencia.



Ejercicio

Representar el siguiente grafo dirigido por medio de su matriz de adyacencia y su lista de adyacencia.

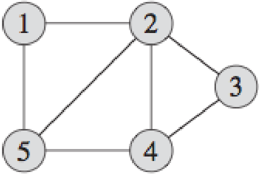




	1	2	3	4	5	6
1	0	1	0	1	0	0
2	0	0	0	0	1	0
3	0	0	0	0	1	1
4	0	1	0	0	0	0
5	0	0	0	1	0	0
6	0	0	0	0	0	1

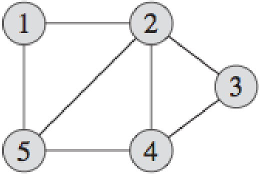
Ejercicio

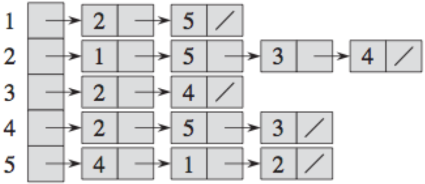
Representar el siguiente grafo no dirigido por medio de su matriz de adyacencia y su lista de adyacencia.



Ejercicio

Representar el siguiente grafo no dirigido por medio de su matriz de adyacencia y su lista de adyacencia.





	1	2	3	4	5
1	0	1	0	0	1
2	1	0	1	1	1
3	0	1	0	1	0
4	0	1	1	0	1
5	1	1	0	1	0

Tarea

Tarea 1.

Escrbir un programa que genere las representaciones como matriz y lista de adyacencia de un grafo **no dirigido** G.
La primera línea de la entrada consiste de dos enteros n el número de nodos y m el número de aristas ($1 \leq n \leq 10$ y $1 \leq m \leq 100$). Las siguientes m líneas tienen cada una dos enteros u y v ($1 \leq u, v \leq n$) que indican que hay una arista que une los nodos u y v.

Tarea 2.

Las representaciones que vimos son para grafos en los que las aristas no tienen pesos. ¿Cómo representarían grafos cuyas aristas tienen pesos?
Pista: Pensar primero cómo lo harían usando la matriz de adyacencia y luego la lista de adyacencia.

La próxima clase

La próxima clase veremos dos métodos para recorrer grafos.

- BFS (Breath First Search)
- DFS (Depth First Search)