Colas de Prioridad

Dora Elisa Alvarado Carrillo

Centro de Investigación en Matemáticas

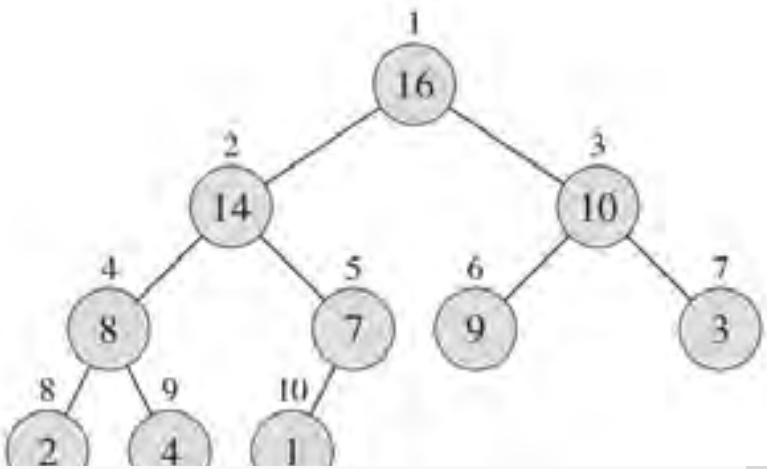
Marzo, 2015

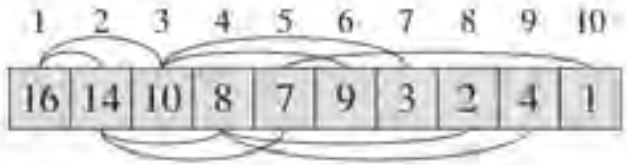
Heap

Heap (binario)

Es un **arreglo** que puede ser visto como un **árbol binario** casi completo. Un arreglo A que representa un **heap** tiene dos atributos:

- **A.length**: Indica el número de elementos en el arreglo
- **A.heap-size**: Representa cuantos elementos válidos en el heap.





Heap

Heap (binario)

Hay dos clases de heaps binarios:

- **max-heap**: Para cada nodo, excepto la raíz, se debe cumplir que $A[Parent(i)] \geq A[i]$.
- **min-heap**: Para cada nodo, excepto la raíz, se debe cumplir que $A[Parent(i)] \leq A[i]$.

Heap

Heap (binario)

```
MAX-HEAPIFY(A, i)
 1  l = LEFT(i)
 2  r = RIGHT(i)
 3  if l ≤ A.heap-size and A[l] > A[i]
 4      largest = l
 5  else largest = i
 6  if r ≤ A.heap-size and A[r] > A[largest]
 7      largest = r
 8  if largest ≠ i
 9      exchange A[i] with A[largest]
10      MAX-HEAPIFY(A, largest)
```

Heap

```
#include <iostream>
#include <algorithm> // make_heap, pop_heap, push_heap, sort_heap
#include <vector>

using namespace std;

int main () {
  int numeros[] = {1,20,14,-5,3};
  vector<int> v(numeros,numeros+5);

  make_heap (v.begin(),v.end());
  cout << "Max inicial heap: " << v.front() << '\n';

  pop_heap (v.begin(),v.end());
  v.pop_back();
  cout << "pop()\n";
  cout << "Max heap despues del pop : " << v.front() << '\n';

  v.push_back(50);
  push_heap(v.begin(),v.end());
  cout << "push(50)\n";
  cout << "Max heap despues del push: " << v.front() << '\n';

  sort_heap (v.begin(),v.end());
  cout << "sort_heap :";
  for (unsigned i=0; i<v.size(); i++)
    cout << ' ' << v[i];

  cout << '\n';
```

Heap

```
Max inicial heap: 20
pop()
Max heap despues del pop : 14
push(50)
Max heap despues del push: 50
sort_heap : -5 1 3 14 50
```

Cola de Prioridad (Priority queue)

Cola de Prioridad

Estructura de datos para mantener un conjunto S de elementos, cada uno con un valor asociado llamado llave o **key**.

- max-priority-queue
- min-priority-queue

Cola de Prioridad (Priority queue)

Operaciones básicas (max-priority-queue)

- **Insert(S,x)** : Inserta el elemento x en el conjunto S.
- **Maximum(S)** : Regresa el elemento de S con la llave mas grande.
- **Extract-max(S)**: Quita y regresa el elemento de S con la llave más grande.
- **Increase-Key(S,x,k)**: Incrementa el valor de la llave del elemento x a un nuevo valor k, que se asume es al menos tan grande como el valor actual de la llave de x.

Cola de Prioridad (Priority queue)

```
#include <iostream>
#include <queue>

using namespace std;

int main () {
    priority_queue <int> pq;        //pq es una cola de prioridad de enteros

    pq.push(2);                     //push 2, 5, 3, 1
    pq.push(5);
    pq.push(3);
    pq.push(1);
    cout<<"pq contiene " << pq.size() << " elementos\n";

    while (!pq.empty()) {
      cout << pq.top() << endl;     //Imprime el elemento con mayor prioridad
      pq.pop();                     //Saca al elemento de mayor prioridad
    }

  return 0;
}
```

Cola de Prioridad (Priority queue)

```
pq contiene 4 elementos
5
3
2
1
```

Referencias

 *http://www.cplusplus.com/reference/algorithm/make_heap*.

 *http://www.cplusplus.com/reference/queue/priority_queue/*.

 Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson.
Introduction to Algorithms.
McGraw-Hill Higher Education, 2nd edition, 2001.