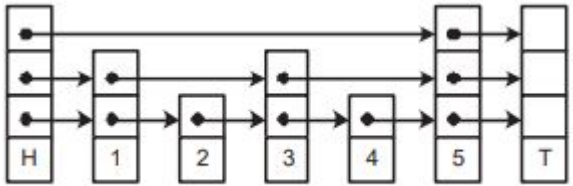
Priority Queue Paralela

Fabián Concha Sifuentes

Implementación del paralelismo

- El algoritmo se basa en el uso de una Skiplist secuencial
- La estructura de datos es básicamente una lista ordenada con atajos distribuidos aleatoriamente para mejorar la búsqueda
- Para asemejarse a una priority queue, los nodos con mayor prioridad se ubican primero en la lista



Priority Queue vs SkipList

- La Priority Queue secuencial cuenta con una complejidad O(N) en su operador push().

- La Skiplist cuenta con una complejidad O(Log N) en su operador push() en el mejor caso.
- Y en el peor caso una complejidad O(N).

Priority Queue, Skip List and Threads

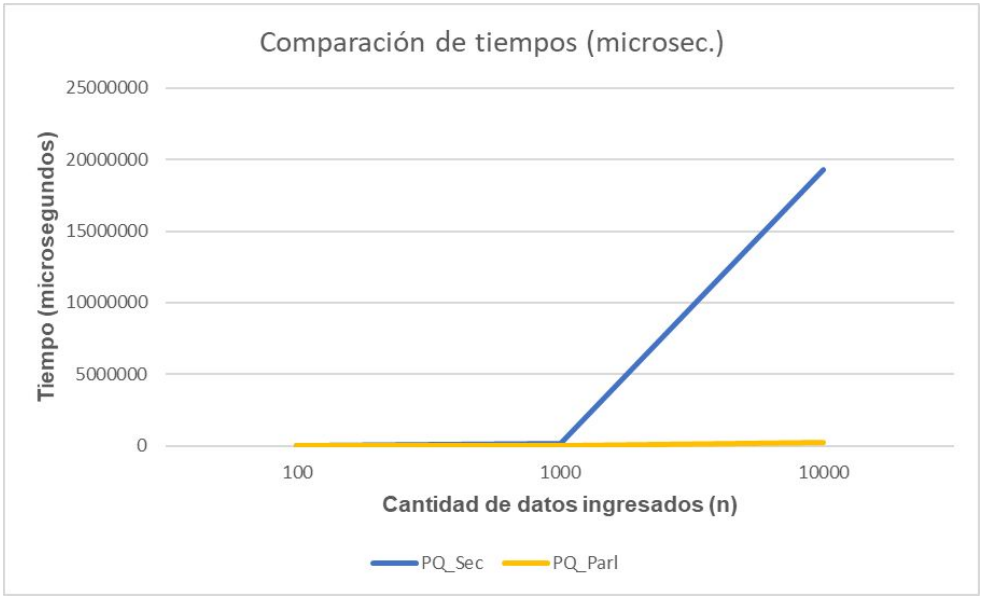
- Crearemos una clase PQ que tendrá todos los atributos de una PriorityQueue convencional, con la siguiente salvedad:
  - En cuanto a las funciones: insert, delete, top, print
  - Haremos uso de las funciones de una Skip List, las cuales declaramos en otra clase
- Esto nos permite manejar los atributos de una Priority Queue y seguir su lógica (Min Heap), pero aprovechar la optimalidad de las funciones de una SkipList
- Se crea una función que permite el uso de threads (4) para llamar a la función insert() y poder realizar varias inserciones al mismo tiempo.

Priority Queue, Skip List and Threads

```
void insert(int n)
{
    srand(time(NULL));
    for (int i = 0; i<n; i++)
    {
        int aux = rand()%(2*n)+1;
        pq.insert_element(aux);
    }
}
```

```
void thr(int n)
{
    int t = 4;
    vector<thread> threads(t);
    for (int i = 0; i < t; i++)
    {
        threads[i] = thread(&PQ::insert, this, n);
    }
    for (int i = 0; i < t; i++)
    {
        threads[i].join();
    }
}
```

Toma de tiempos en inserción de datos PQ Sec vs. PQ Parl



- PQ secuencial

```
Insert 100 elem.: 88 microsec.
Insert 1000 elem.: 3034 microsec.
Insert 10 000 elem.: 153761 microsec.
Insert 100 000 elem.: 19264884 microsec.
```

- PQ Paralela

```
Insert 100 elem.: 567 microsec.
Insert 1000 elem.: 810 microsec.
Insert 10 000 elem.: 8964 microsec.
Insert 100 000 elem.: 270650 microsec.
Insert 1 000 000 elem.: 3172205 microsec.
```