Estructuras de Datos y Algoritmos II

Tarea 1

Julián Rodríguez Isiordia Licenciatura en Computación Matemática

Se decidió utilizar un mapa para poder mantener un orden respecto a los elementos y sus índices. Hubo unos problemas cuando no se actualizaban los índices en las operaciones de borrado $(eraseypop)$, lo cual era más fácil mantener usando un map. La condición que deben cumplir $TPriority$ y $TKey$ es tener un orden con el (o los) operador(es) $<, >$.

Complejidad

Primero veamos las complejidades de las operaciones.

1. **pop** Esta operación tiene complejidad $O(\log N)$, donde N representa la cantidad de nodos, pues ocupa verificar la propiedad del heap una vez que se ha hecho la extracción usando la función $heapifyDown$ la cual tiene complejidad $O(\log N)$.
2. **top** Tiene complejidad $O(1)$ pues simplemente regresa la pareja del primer nodo.
3. **isInserted** Tiene complejidad $O(\log N)$, en el caso promedio y $O(N)$, en el peor caso pues ocupa la función find, que tiene las mismas complejidades.
4. **getSize** Tiene complejidad $O(1)$, pues simplemente devuelve la cantidad de nodos en el heap.
5. **erase** Tiene complejidad $O(\log N)$, pues ocupa verificar la propiedad del heap una vez que se ha hecho la extracción usando la función $heapifyUp$ o $heapifyDown$ las cuales tiene complejidad $O(\log N)$.
6. **insertOrUpdate** Tiene complejidad $O(\log N)$, pues una vez que se realiza la actualización, ocupa verificar la propiedad del heap la función $heapifyUp$, o $heapifyDown$ en algunos casos de actualización, las cuales tiene complejidad $O(\log N)$.

Con lo anterior en mente, al realizar P operaciones de inserción, borrado o actualización, tenemos que la complejidad es $O(P \log N)$.