Priority_queue_V1

```
1 void priority_queue<T>::push(const T & t){
2       V.push_back(t);
3       if(size() == 1)
4               mayor = 0;
5       if(t > V[mayor])
6               mayor = V.size()-1;
7}

….
8 bool priority_queue<T>::empty() const{
9       if (V.size() == 0)
10              return true;
11      else
12              return false;
13}

….
14 const T & priority_queue<T>::top( ) const{
15      if(!empty()){
16              assert(cheq_rep());
17              return V[mayor];
18      }
19}

….
20 void priority_queue<T>::pop( ){
21      if(!empty()){
22              V.erase(V.begin()+mayor);
23              for(int i = 0; i < V.size(); i++){
24                      if(V[mayor] < V[i] || i == 0) mayor = i;
25              }
26      }
27}

…

28void ordenar(vector<string> & V, int tama){

29      priority_queue<string> aux;
30      int pos;
31      for (int i=0;i<tama; i++)
32              aux.push(V[i]);

33      pos = tama-1;
34      while (!aux.empty()) {
35              V[pos]=aux.top();
36              aux.pop();
37              pos--;
38      }
40}
```

Análisis teórico:

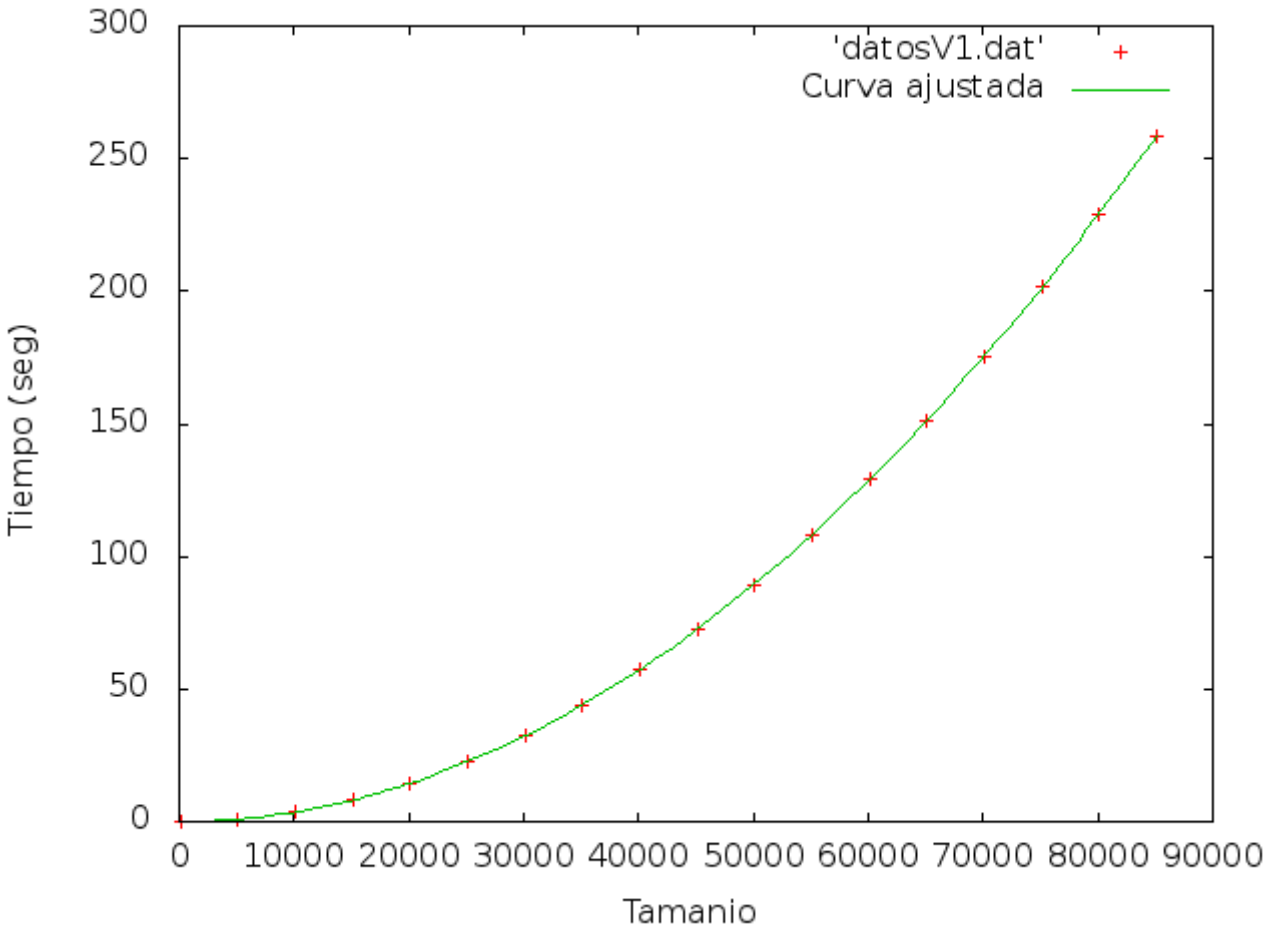
El método push tiene una eficiencia de orden O(1).
El método empty tiene una eficiencia de orden O(1).
El método top tiene una eficiencia de orden O(1).
El método pop tiene una eficiencia de orden O(n).

El método ordenar:

- Las lineas 31 a 32 son las pertenecientes al bucle for que ejecutan un push con O(1) n veces por tanto dichas lineas están acotadas por O(n).
- Las lineas 35 a 37 están acotadas por O(max(O(1), O(n), O(1)) = O(n).
- Las lineas 34 a 38 son las pertenecientes al bucle while que ejecutan las lineas 35 a 37 n veces por tanto las lineas 34 a 38 están acotadas por $O(n^2)$.
- El orden de eficiencia conjunto (linea 29, linea 30, lineas de 31 a 32, linea 33 y lineas de 34 a 38) es de O(max(O(1), O(1), O(n), O(1), $O(n^2)$) = $O(n^2)$.

Por tanto el método ordenar tiene una eficiencia teórica de orden $O(n^2)$.

Análisis empírico:



La curva que se ha ajustado a los puntos de los datos es una recta $f(x) = a * x^2$; donde a = 3.57056 $*10^{-08}$.
Con un error en el ajuste del 0.02839%

Por tanto el análisis empírico se ajusta a lo descrito por el análisis teórico.

Priority_queue_V2

```
1 void priority_queue<T>::push(const T & t){
2       if(empty()) // Si está vacio lo añade directamente
3               V.push_back(t);
4       else{
5               int i= V.size() -1;
6               while(i>0 && V[i]<t){
7                       i--;
8               }
9               if(V[i] > t)
10                      V.insert(V.begin()+i+1,t);
11              else
12                      V.insert(V.begin()+i,t);
13      }
14}
….
15 bool priority_queue<T>::empty() const{
16      if (V.size() == 0)
17              return true;
18      else
19              return false;
20}
….
21 const T & priority_queue<T>::top( ) const{
22      if(!empty()){
23              assert(cheq_rep());
24              return V[0];
25      }
26}
….

27 void priority_queue<T>::pop( ){
28      if(!empty())
29              V.erase(V.begin());
30}
….
31 void ordenar(vector<string> & V, int tama){

32 priority_queue<string> aux;
33 int pos;

34 for (int i=0;i<tama; i++)
35   aux.push(V[i]);

36 pos = tama-1;
37 while (!aux.empty()) {
38 V[pos]=aux.top();
39aux.pop();
40   pos--;
41}
}
```

Análisis teórico:

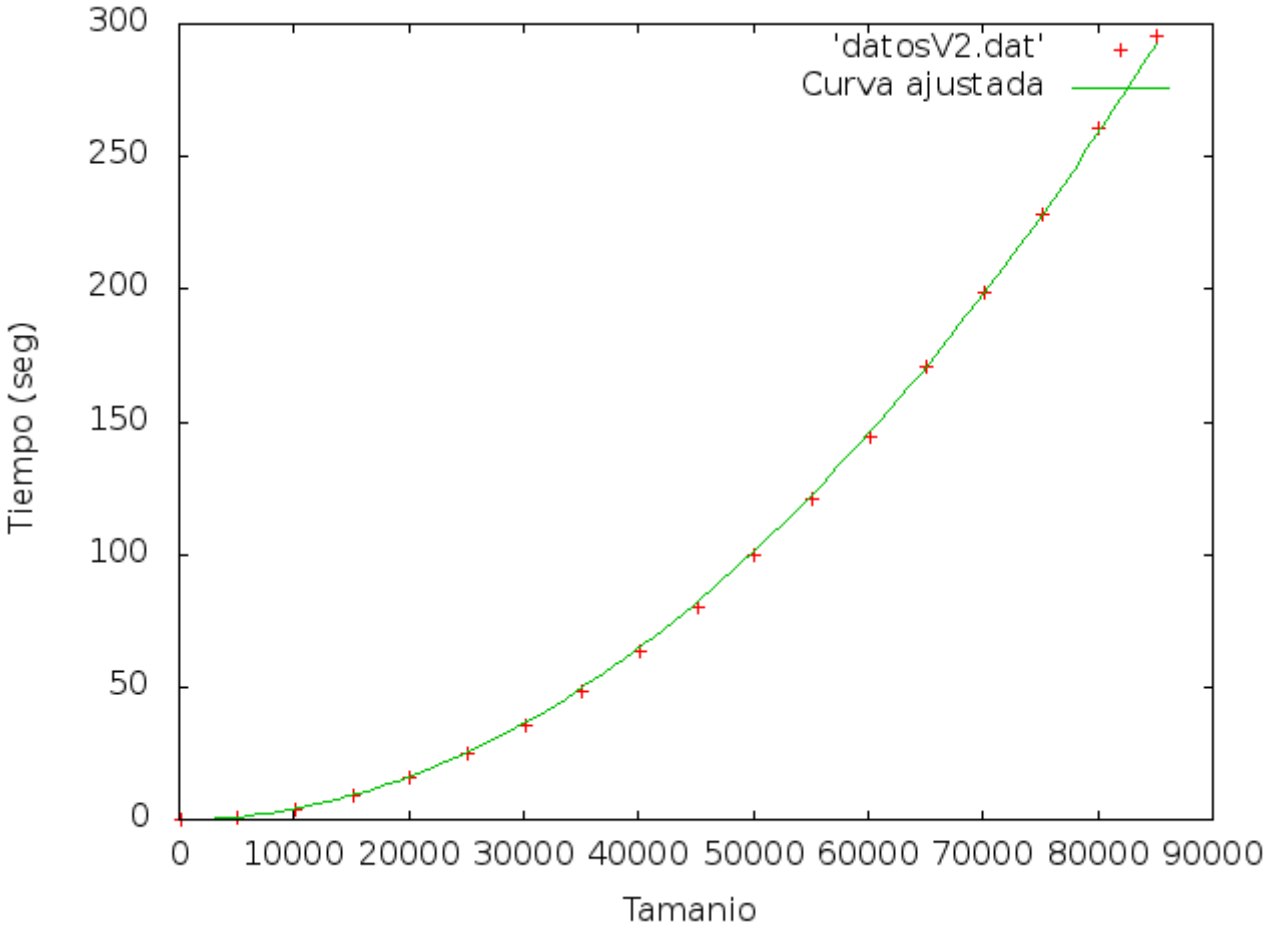
El método push tiene una eficiencia de orden O(n).
El método empty tiene una eficiencia de orden O(1).
El método top tiene una eficiencia de orden O(1).
El método pop tiene una eficiencia de orden O(1).

El método ordenar:

- Las lineas 34 a 35 son las pertenecientes al bucle for que ejecutan un push con O(n) n veces por tanto dichas lineas están acotadas por $O(n^2)$.
- Las lineas 38 a 40 están acotadas por O(max(O(1), O(1), O(1)) = O(1).
- Las lineas 37 a 41 son las pertenecientes al bucle while que ejecutan las lineas 38 a 40 n veces por tanto las lineas 34 a 38 están acotadas por O(n).
- El orden de eficiencia conjunto (linea 32, linea 33, lineas de 34 a 35, linea 36 y lineas de 37 a 41) es de $O(max(O(1), O(1), O(n^2), O(1), O(n)) = O(n^2)$.

Por tanto el método ordenar tiene una eficiencia teórica de orden $O(n^2)$.

Análisis empírico:



La curva que se ha ajustado a los puntos de los datos es una recta $f(x) = a * x^2$; donde a = 4.04217 $*10^{-08}$.
Con un error en el ajuste del 0.2218%

Por tanto el análisis empírico se ajusta a lo descrito por el análisis teórico.

Conclusión: aunque el método ordenar usando Priority_queue_V1 y Priority_queue_V2 tenga una eficiencia teórica del mismo orden ($O(n^2)$) en este ejemplo de uso concreto la eficiencia empírica usando Priority_queue_V1 es un poco mejor ya que ajustando los resultados a una $f(x) = a * x^2$ la constante oculta (a) es $3.57056 * 10^{-08}$ mientras que usando Priority_queue_V2 es $4.04217 * 10^{-08}$. No es una diferencia muy significativa, por lo que cuando nos interese una representación de una cola con prioridad con vector interno ordenado podemos recurrir a la Priority_queue_V2 sin temer por que la eficiencia vaya a empeorar de forma relevante.