Colas de Prioridades en Memoria Secundaria

Estructuras de Datos Conocidas

	find min	insertar	borrarMin	Espacio
lista enlazada ord	1	n	1	2n
desord	n	1	n	2n
circ				2n
heap	1	lg n	lg n	n
∀ diccionarios (--)				
Fibonacci heap (1 - -)				≥3n

① Adaptar Barbol o vEB arbol

$O(\log_B n)$ FindMin, BorrarMin, Insertar

Diccionario Ordenado (llave, dato)	Cola de Prioridad (llave, dato)
Busqueda {exacta, find next	EncuentraMinimo (FindMin)
Insercion	~~EncuentraMaximo~~
Borrado	Insertar
	BorrarMinimo esta vacilla
	~~Obtener~~

Th Todos diccionarios soportan los metodos de Colas de Prioridades

Proof

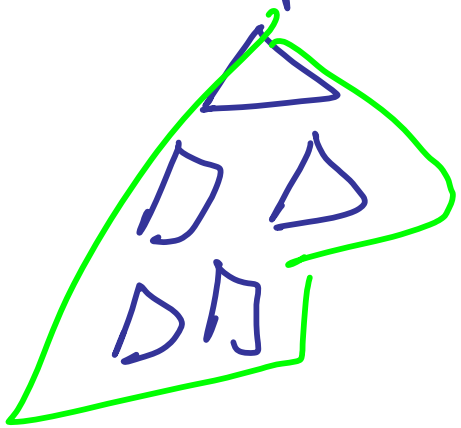
- para iniciar una cola de prioridad
  iniciar un diccionario con $\{-\infty\}$
- para insertar (prioridad, dato)
  insertar( ) en diccionario.
- para buscar el minimo
  findNext($-\infty$)
- para borrar el minimo
  buscar + borrarlo

esta vacilla findNext($-\infty$)

QED

② Se puede hacer mejor?

- Espacio? Si!



$\lceil n/B \rceil$ paginas organizadas en un heap de paginas

en cada pagina, un heap con B elementos (falta la ultima)

- $\lceil n/B \rceil B \leq n + B$ espacio
- $O(\log_B n)$

(quizas mismo con vEB?)

- Tiempo? Se puede mostrar $\Omega(\log_B(n))$?

Si! Con una reduccion, en el modelo de comparacion.

Si existiera una cola de prioridad con operaciones realizadas en $o(\log_B n)$

Se podria ordenar en tiempo $o(n \log_B n)$

⚠ la reduccion a diccionario NO permite de deducir una cota inferior para colas de prioridades (al reves)

Ordenar en Memoria Secundaria

B = tamaño pagina $\quad N = \lceil n/B \rceil$ = # paginas a ordenar

n = # datos $\quad M = \lceil m/B \rceil$ = # paginas en memoria

m = tamaño memoria principal

- ordenar cada pagina
  - $O(n/B)$ accesses
  - $\frac{n}{B} B \lg B = n \lg B$ comparaciones

- mergesort sobre las paginas
  - poner $M-1$ paginas en memoria principal
  - buscar linealmente el minimo de los $M-1$ minimos
  - guardar en una pagina (si llena, escribirla)

+ recursion (--->)

$\Rightarrow O(N \log_m N)$ accessos

$\Rightarrow O\left(\frac{n}{B} \frac{\log n/B}{\log m/B}\right)$

¿Cuál sería la complejidad de ordenar con un B-árbol? ¿Y con un B-heap? $O\left(n \frac{\log n}{\log B}\right)$ accesos