Unidad 5: Algoritmos de memoria externa

Ordenamiento

Gibran Fuentes Pineda

Junio 2020

Ordenamiento por distribución para memoria externa (1)

- Se usan $S - 1$ elementos $e_1, \ldots e_{S-1}$ para dividir problema en subarreglos S (cubetas)
- El i-ésimo subarreglo consiste en todos los elementos con un valor entre $[e_{i-1}, e_i)$
- División, ordenamiento y concatenación recursivos de los subarreglos hasta tener todos los elementos ordenados
- La proceso recursivo termina cuando el subarreglo es de B elementos (cabe en un bloque de tamaño B)

Ordenamiento por distribución para memoria externa (2)

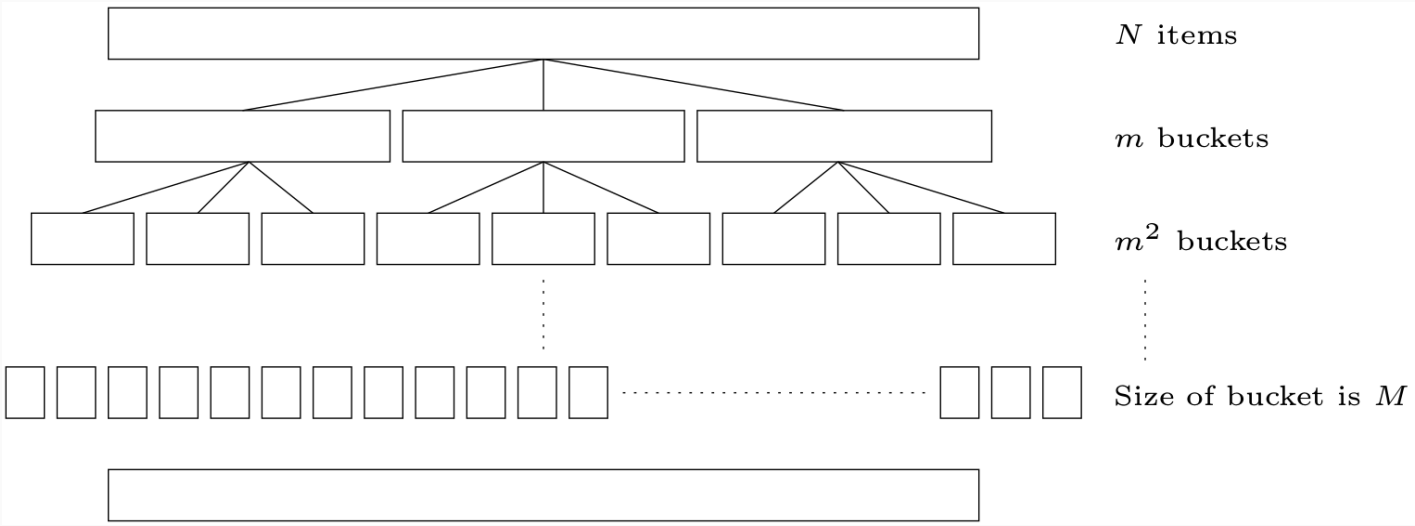

Imagen tomada de Lars Arge: *External-Memory Algorithms with Applications in Geographic Information Systems*, 1996.

Ordenamiento por distribución para memoria externa (3)

- Retos
  - Encontrar elementos que generen subarreglos de tamaño similar
  - Cuando $D > 1$, balancear la carga entre los discos
  - Estrategia de escritura de subarreglos en disco

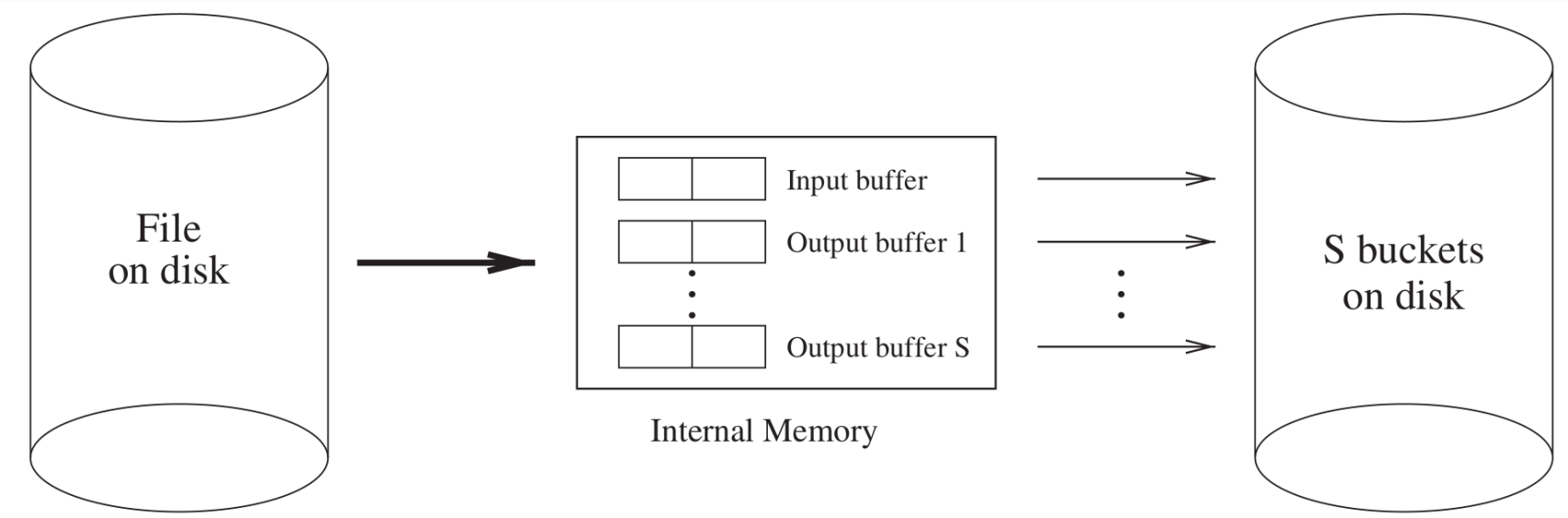

Imagen tomada de Vitter: *Algorithms and Data Structures for External Memory*, 2008.

Ordenamiento por mezcla en memoria interna (1)

- Algoritmo de ordenamiento basado comparaciones diseñado por John von Neumann

- Paradigma *divide y vencerás*
  1. Divide recursivamente el arreglo de N elementos hasta obtener N subarreglos de un solo elemento
  2. Ordenar y mezclar repetidamente los subarreglos ordenados hasta que quede un solo arreglo

- Complejidad $O(N \log N)$: $\log N$ niveles y N comparaciones en cada nivel

Ordenamiento por mezcla en memoria interna (2)

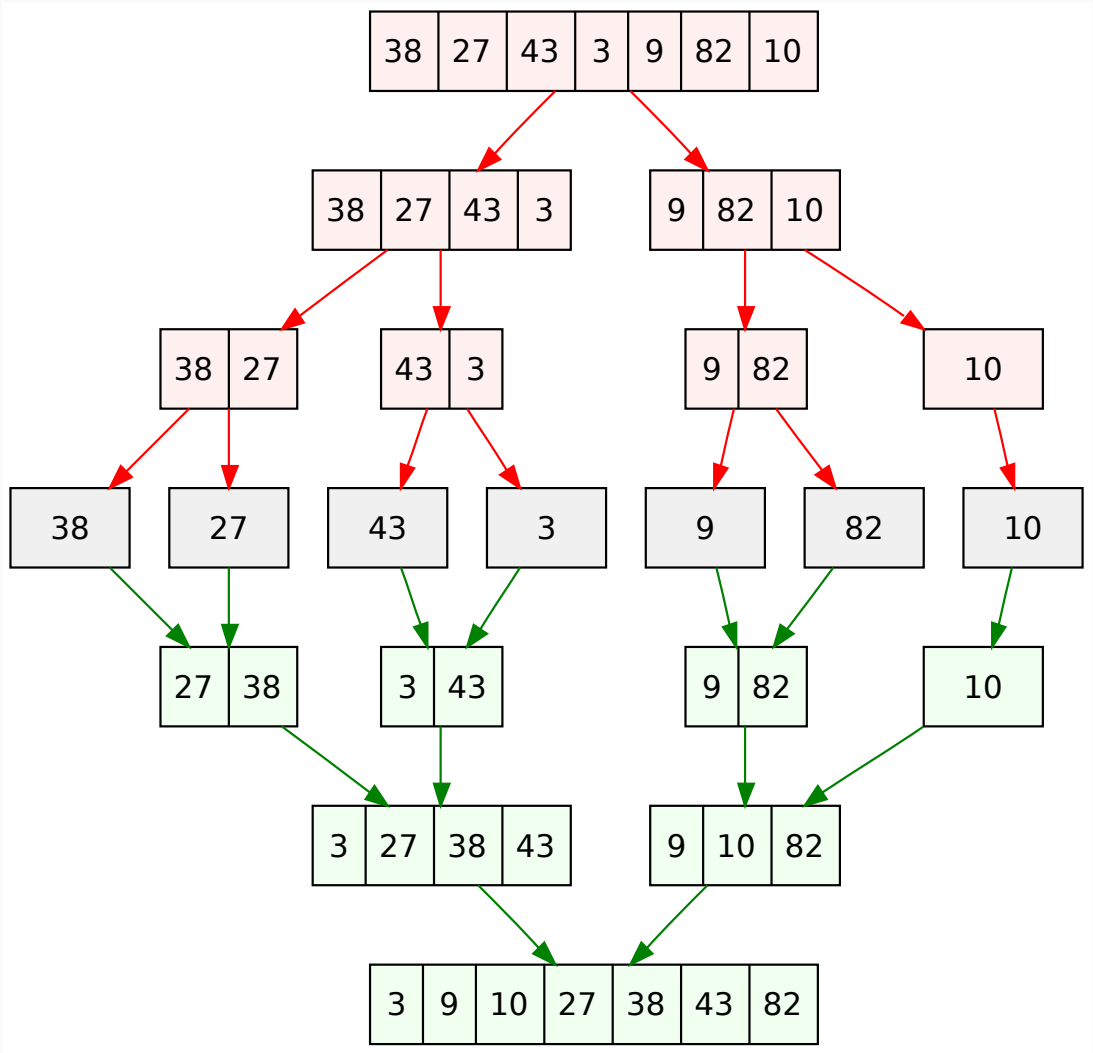

Imagen tomada de Wikipedia (Merge Sort)

Ordenamiento por mezcla en memoria externa: división

- Divide arreglo en N/M subarreglos de tamaño M
- Ordena cada subarreglo de forma independiente (trabajo)
- $2N/B$ transferencias, $2M/B$ por subarreglo
- $N \log M$ comparaciones, $M \log M$ por subarreglo

Ordenamiento por mezcla en memoria externa: mezcla

- Mezcla 2 trabajos R y S de tamaño L en uno solo T de tamaño $2L$ (conocido como *mezcla de 2 caminos*)
  1. Carga primeros bloques $\hat{R}$ y $\hat{S}$ de R y S
  2. Aloja el primer bloque $\hat{T}$ de T
  3. Mientras haya elementos en R y S
     a. Mezcla todos los posibles elementos de $\hat{R}$ y $\hat{S}$ en $\hat{T}$
     b. Si ya no hay elementos en $\hat{R}$ o $\hat{S}$, carga un nuevo bloque
     c. Si $\hat{T}$ se llena, cópialo a T
  4. Copia el resto de elementos de R o S a T

- Transferencias: $\frac{2L}{B}$ lecturas, $\frac{2L}{B}$ escrituras y $2L$ comparaciones

Mezcla de K caminos

- Mezcla K trabajos de forma eficiente (por ej. con montículos mínimos)
- $\frac{2KL}{B}$ transferencias por mezcla
- Maximiza K para reducir transferencias
  - (K lecturas + 1 escritura) B = M
  - Total de transferencias $O\left(\frac{N}{B}\log_{M/B}\frac{N}{B}\right)$

Ordenamiento por mezcla en memoria externa (2)

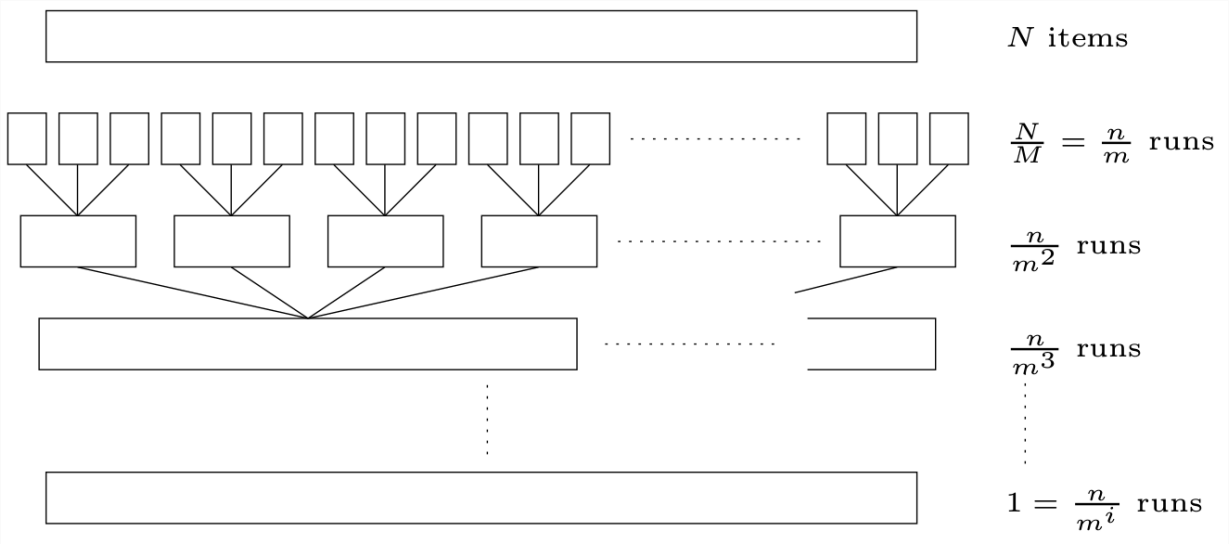

Imagen tomada de Lars Arge: *External-Memory Algorithms with Applications in Geographic Information Systems*, 1996.

Ordenamiento por mezcla en memoria externa (3)

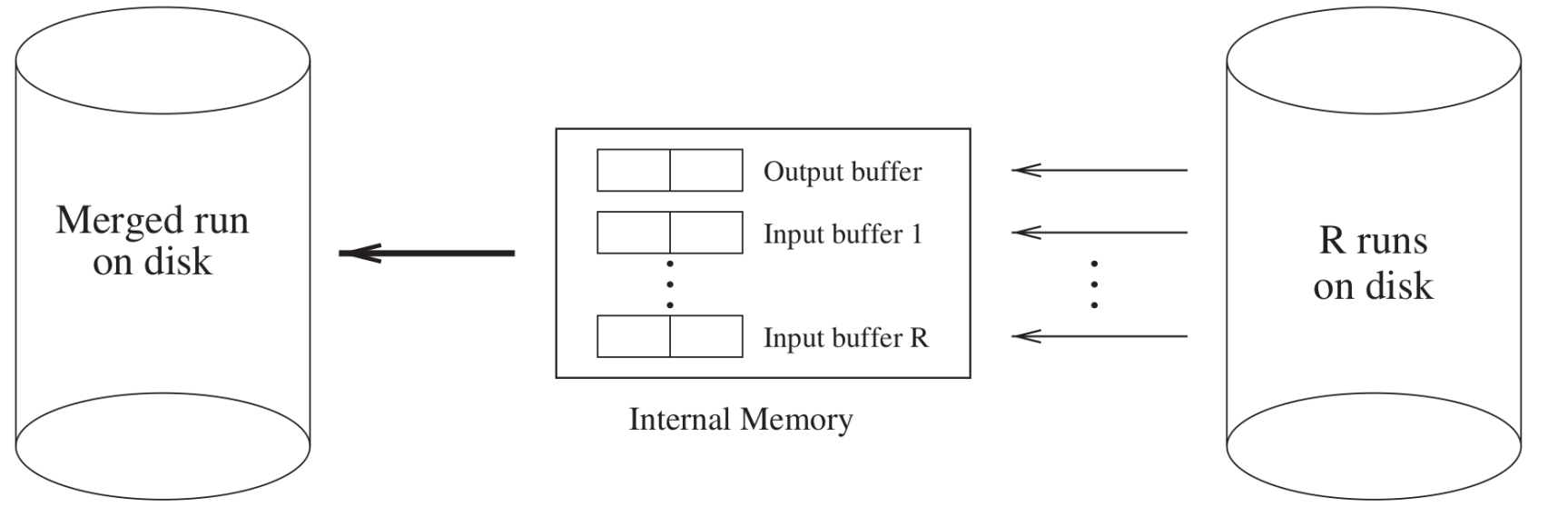

Imagen tomada de Vitter: *Algorithms and Data Structures for External Memory*, 2008.

Ordenamiento en modelo de inconsciente de caché

- Es posible usar el ordenamiento por mezcla de 2 caminos pero no es muy eficiente

- Alternativa más eficiente: ordenamiento de embudo
  1. Divide el arreglo en $K = N^{1/3}$ subarreglos contiguos (cada uno de tamaño $N/K = N^{2/3}$) y los ordena de forma recursiva
  2. Mezcla los K subarreglos ordenados de forma eficiente usando K-embudos[1]

- Ordenamiento de embudo requiere $O(\frac{N}{B} \log_{M/B} \frac{N}{B})$ transferencias

[1] Demaine: *Cache-Oblivious Algorithms and Data Structures*, 2002