Propuesta para trabajar en clase

30/08/2019

- Elegí alguna de las propuestas de enunciado siguientes, según el método que te tocó investigar.
- Armá grupos de 3 personas
- Creá los casos de Test y comenzá a programar tu solución.

1- **Ordenamientos A** - Suma dígitos: (Bubble-Sort, **Bucket-Sort**, Quick-Sort)
Ordenen una lista de numeros enteros no por su valor sino por cuanto da la suma de sus digitos, desempatando en caso de coincidencias por el orden natural de los mismos.
**Necesita casos de tests especificos. Bucket para agrupar por el valor de la suma + quick sort para cada bucket. Después hacen una pasada que itera por los buckets listando todos los elementos ordenados en cada bucket. (¿Veriamos mal que lo implementen armandolo con TreeMap<Integer, TreeSet<Integer>>?

2. **Ordenamientos B** - "Chef and Pick Digits": (**Counting Sort**, Quick Sort, Selection Sort)
https://www.codechef.com/SEPT17/problems/CHEFPDIG
A Chef le gusta jugar con números grandes. Hoy, el tiene un entero positivo verdaderamente gigantesco: El puede seleccionar dos dígitos cualesquiera de este número (Los dígitos pueden ser iguales pero sus posiciones deben ser distintas) y los ordena en ambos sentidos posibles. Por cada uno de estos entidos, el crea un número de dos dígitos. Luego el tomará el caracter ASCII cuyo valor sea igual al número de dos dígitos considerado, esto es: el número 65 corresponde a 'A', el 66 a 'B' y sucesivamente, el 90 a 'Z'. Chef estáinteresado solo en encontrar cuales letras mayúsculas puede tomar de esta manera.

Ordenen una lista de números enteros entre 1 y 100.000. La lista puede tener hasta 10.000.000 de elementos.
(Desafío no evaluativo: https://www.codechef.com/problems/TSORT)
**X¿Demasiado facilito? Counting Sort, derechisimo. Alternativa algo más complicada: https://www.codechef.com/SEPT17/problems/CHEFPDIG

2 - **Ordenamientos C** - "Trabajo muy duro, como un esclavo": (Merge sort, **Counting sort**, Quick sort)
Juan y Carla tienen cada uno ordenadas las tareas que deben realizar en el trabajo la próxima semana ordenadas por prioridad.
No es vital que se cumplan las fechas pero si el orden de las mismas. Por razones de fuerza mayor, Juan no puede ir a trabajar, por lo que el jefe de ambos le pide a Carla que avance lo más posible en las tareas de ambos, para esto Carla necesita organizar las tareas, respetando las prioridades que ambos tenian

establecidas.
Ayuden a Carla determinando el orden TOTAL de todas las tareas.
**¿Demasiado facilito? La parte de la mezcla del mergesort haria el trabajo

3. **Ordenamientos D** - TimeStamps: (Selection sort, Heap sort, **Radix sort**)
Como todos sabemos existen aplicaciones utilizadas por miles de usuarios en simultáneo, tales como servidores de correo electrónico, redes sociales, mensajería instantánea, reproducción de multimedia por streaming, etcétera.
Suponiendo que ustedes se presenten a entrevistas para trabajar en una empresa que provea alguno de estos servicios como ingenieros de software, es probable que enfrenten entrevistas técnicas con preguntas del estilo: "Dados los siguientes registros de accesos a la página tal, para los cuales queda registrada la fecha y hora además del usuario que accedió a la página.

Suponiendo que hay millones de registros "YYYY:MM:DD-HH:MM:SS--Usuario" y distintas listas de usuarios ¿Cómo ordenaría estos registros para resolver consultas del estilo "obtenga los últimos 200 registros de los usuarios de cada lista"?
** i) Radix-Sort derecho para las fecha-horas para. Después iterar de último a primero y levantar los ultimos registros de los usuarios en cada lista.
ii) Tener una función que les calcule un numero en funcion de la fecha-hora (cantidad de segundos desde una fecha específica) y con un heap salen andando.

2 - 4 - **Ordenamientos E** - ¿Es permutación? (Heap Sort, **Tree Sort, Counting Sort**)
Dados dos arreglos de hasta un millón de elementos cada uno, determinar si uno es permutación del otro.
(*) Probablemente para cada subproblema usen un ordenamiento distinto.
i. ¿Cómo lo resuelven si saben que los arreglos solo contienen letras minúsculas?
** i. Con Counting Sort sale lineal (O(n)+O(m)+O(1) de la iteracion por las letras chequeando si coinciden las cantidades). con Heap en O(n*log(n))

2 - 4 - **Ordenamientos F** - ¿Es permutación? (Heap Sort, **Tree Sort**, **Counting Sort**)
Dados dos arreglos de hasta un millón de elementos cada uno, determinar si uno es permutación del otro.
(*) Probablemente para cada subproblema usen un ordenamiento distinto.
ii. ¿Si en vez de tener solo letras minusculas pudieran tener cualquier valor entre 1 y 10^9?
ii. El O(1) del caso anterior ahora no se puede hacer por la cantidad de valores, al menos, no derecho...
El Heap acá queda bastante bien. Salvo que se implemente Bags (Multisets) con TreeMap o algo asi, que seria un hibrido de Counting Sort con TreeSort.