

Lab 6 – 21st Jan. 2017

Topics – Bucket / Bin Sorting and Variants

1. Multi-key Sorting of Sparse Lists:

- Consider a list L_s of points in 2-dimensional integer grid i.e. points represented as pairs $\langle x, y \rangle$ of integers:
 - Assume that the points are unique.
 - Also assume the range of the dimensions of the grid are known i.e. for any point $\langle x, y \rangle$ it is known that $x_{Lo} \leq x \leq x_{Hi}$, and $y_{Lo} \leq y \leq y_{Hi}$.
- The list L_s is said to be sparse if the size of L_s is $O(m+n)$ where $m=x_{Hi}-x_{Lo}$ and $n=y_{Hi}-y_{Lo}$; L_s is said to be dense otherwise.
- Sorting of such a list L_s of Points can be done by bucketing based on the x-value and then inserting into a sorted list which is sorted based on the y-value (or alternatively by bucketing based on the y-value and then inserting into a sorted list that is sorted based on the x-value).
- For e.g. a point $\langle 3, 17 \rangle$
 - is inserted – in the right order – into a linked list that is the x-bucket indexed 3.
 - or alternatively, is inserted - in the right order – into a linked list that is the y-bucket indexed 17.
- Implement this sorting scheme using an array of linked lists.
- The sorted list must be copied back into L_s .
- Note that the sorting algorithm should accept the ranges (i.e. $x_{Lo}, x_{Hi}, y_{Lo}, y_{Hi}$) as parameters and allocate the array dynamically. Bucketing is done on the basis of x-values if the x-range is larger than the y-range (and vice-versa). [Hint: This leaves the insertions in each bucket to be done on shorter lists.]

Parameterize your code based on the following functions.

Key	Function	Input Format	Description
0	readData	0 M xLo xHi yLo yHi	Indicates that next M lines will contain data to be read. Each line will contain a tuple $\langle X Y \rangle$ separated by space. You must create a 2-dimensional array that stores these read points. x_{Lo} , x_{Hi} , y_{Lo} and y_{Hi} are the values as explained before.
1	SortSparseLists	1	Sorts the sparse list as explained by the above algorithm and writes back the sorted tuples to the 2-dimensional array that contained the input read. Also, this function must print the sorted list.

Test Case 1:

Input	Output
0 20 89 99 85 95	90 86
92 92	90 92
98 90	91 85
93 87	91 93
93 89	92 85

91 93	92 92
93 90	92 93
91 85	93 86
94 92	93 87
93 88	93 88
92 93	93 89
96 94	93 90
94 91	93 92
93 86	94 91
93 92	94 92
90 92	96 94
90 86	97 85
92 85	97 86
97 85	97 88
97 88	98 90
97 86	
1	

2. Multi-key Sorting of Dense Lists:

- If L_s is dense, the points in x-bucket can in turn be bucketed, i.e. a point $\langle 3, 17 \rangle$ in a space ranging from $\langle 0, 0 \rangle$ to $\langle 40, 100 \rangle$ for instance, is stored in a y-bucket indexed 17 within a x-bucket indexed 3.
- Implement this sorting scheme using an array of arrays for buckets.
- The sorted list must be copied back into L_s .
- Note that the sorting algorithm should accept the ranges (i.e. xLo, xHi, yLo, yHi) as parameters and allocate arrays dynamically.

Use the above code, and create an additional function as per the following table.

Key	Function	Input Format	Description
2	SortDenseLists	2	Sorts the dense list as explained by the above algorithm and writes back the sorted tuples to the 2-dimensional array that contained the input read. Also, this function must print the sorted list.

Test Case 1:

Input	Output
0 11 1 6 7 12	1 7
4 10	1 9
4 7	1 10
3 8	2 10
3 10	2 11
3 9	3 8
4 8	3 9
2 11	3 10
1 10	4 7

1 7	4 8
1 9	4 10
2 10	
2	

3. Instrument the two procedures (i.e. your solutions to 1 and 3) to measure the time taken and heap space used:
 - a. Measure the time taken and space used by the two procedures for different input lists (some of which are
 - sparse, i.e. of size from $O(k)$ for $k \ll m$ and $k \ll n$ to size $c \cdot (m+n)$ for a small constant c , where m and n are as defined in 1.
 - and others are
 - dense, i.e. of size from $O((m+n) \cdot \log(m+n))$ to $O(m \cdot n)$.
 - b. Based on these measurements identify ranges of input sizes for which your solution in 1 is better than the other or vice-versa; and ranges where neither is superior.