CS 3345. Data structures and algorithm analysis

Project 3: Multi-dimensional search

Due: 11:59 PM, Sunday, Nov 12.

Project Description
Implement the following operations. Starter code is provided.
Do not change the name of the class.
Do not change the signatures of public methods in the starter code.
Few testcases and verbose output for some of the testcases are also provided.
Check 'P3' folder in the shared box folder.

Multi-dimensional search: Consider the web site of a seller like Amazon. They carry tens of thousands of products, and each product has many attributes (Name, Size, Description, Keywords, Manufacturer, Price, etc.). The search engine allows users to specify attributes of products that they are seeking, and shows products that have most of those attributes. To make search efficient, the data is organized using appropriate data structures, such as balanced trees. But, if products are organized by Name, how can search by price implemented efficiently? The solution, called indexing in databases, is to create a new set of references to the objects for each search field, and organize them to implement search operations on that field efficiently. As the objects change, these access structures have to be kept consistent.

In this project, each object has 3 attributes: id (int), description (zero or more ints), and price (int). The following operations are supported:

a. Insert(id,price,list): insert a new item whose description is given in the list. If an entry with the same id already exists, then its description and price are replaced by the new values, unless list is null or empty, in which case, just the price is updated. Returns 1 if the item is new, and 0 otherwise.

b. Find(id): return price of item with given id (or 0, if not found).

c. Delete(id): delete item from storage. Returns the sum of the ints that are in the description of the item deleted (or 0, if such an id did not exist).

d. FindMinPrice(n): given an integer, find items whose description contains that number (exact match with one of the ints in the item's description), and return lowest price of those items. Return 0 if there is no such item.

e. FindMaxPrice(n): given an integer, find items whose description contains that number, and return highest price of those items. Return 0 if there is no such item.

f. FindPriceRange(n,low,high): given int n, find the number of items whose description contains n, and in addition, their prices fall within the given range, [low, high].

g. RemoveNames(id, list): Remove elements of list from the description of id. It is possible that some of the items in the list are not in the id's description. Return the sum of the numbers that are actually deleted from the description of id. Return 0 if there is no such id.

Implement the operations using data structures that are best suited for the problem.

Input specification:
Initially, the store is empty, and there are no items. The input contains a sequence of lines (use test sets with millions of lines). Lines starting with "#" are comments. Other lines have one operation per line: name of the operation, followed by parameters needed for that operation (separated by spaces). Lines with Insert operation will have a "0" at the end, that is not part of the name. The output is a single number, which is the sum of the following values obtained by the algorithm as it processes the input.

Sample input:

```
Insert 22 19 475 1238 9742 0
# New item with id=22, price="$19", name="475 1238 9742"
# Return: 1
#
Insert 12 96 44 109 0
# Second item with id=12, price="96", name="44 109"
# Return: 1
#
Insert 37 47 109 475 694 88 0
# Another item with id=37, price="47", name="109 475 694 88"
# Return: 1
```

#
FindMaxPrice 475
Return: 47 (id of items considered: 22, 37)
#
Delete 37
Return: 1366 (=109+475+694+88)
#
FindMaxPrice 475
Return: 19 (id of items considered: 22)
#
End

Output:
1435

Insert 849 6 3320 2523 810 1488 589 2758 1712 2572 2340 614 2063 1187 843 1004 501 662 2408 1172 2599 455 856 1648 801 0
Insert 764 11 2834 1328 2482 1409 969 168 2052 1346 3247 1688 3397 340 228 1490 842 2765 2283 712 2167 3336 2485 1272 791 492 1665 3430 1892 2971 283 752 1101 1643 1685 2697 1364 3156 2044 3418 2059 1191 2169 3281 1045 711 1283 2409 2067 1892 2120 2093 2183 1780 2912 2259 180 581 2597 546 3032 1462 2900 3248 1800 2193 3327 669 1319 3275 3067 819 1561 2496 1003 2553 945 0
Find 849
Insert 369 3 1733 3049 2623 207 105 644 2111 963 289 199 3240 222 2322 866 1639 3085 2653 1988 1528 2402 2965 2126 888 1250 490 942 2788 108 2321 2258 825 3345 433 1143 2020 2436 3262 861 3077 2341 2788 2755 733 2397 2807 1296 1291 826 253 2510 926 1750 155 600 3038 831 625 3218 110 950 2873 3192 2546 3272 2925 0
Insert 292 8 1483 1317 317 1945 2960 2290 2881 1251 2332 3026 1930 1645 2188 3075 397 2039 2298 2229 3257 233 2792 263 1745 2494 986 2279 1744 2284 1100 1173 2180 894 1073 783 420 1557 1825 3363 211 2621 3176 201 2442 2712 2824 557 1982 718 2384 2518 2671 3138 2198 2315 2161 1462 418 923 1120 2960 0
Insert 393 4 1184 1874 497 400 2721 2799 2416 625 2028 2806 1449 1861 3363 1717 529 142 504 2497 634 916 2484 2735 780 2886 291 1085 2295 1965 703 2069 239 703 1520 3056 511 2163 825 1742 704 2044 2831 1639 2945 2462 735 1880 1380 2418 3331 3359 854 2012 3210 662 1359 2565 2058 1178 2039 521 0
Insert 973 2 1131 3319 2573 2487 2995 1013 3112 3366 2623 2399 169 806 616 1720 1533 669 1331 1136 1807 127 1018 1601 3309 652 968 1520 340 1336 273 665 2071 1374 1862 550 1229 0
Find 292
Insert 535 6 820 3098 2051 3119 2808 2881 1774 674 1983 1990 985 1852 3014 206 1912 2692 2275 2569 3202 2231 764 309 1116 440 2135 1304 2858 1197 3427 939 109 529 1192 407 1534 227 1765 111 1579 1202 3257 2368 150 0
Find 764
End

Output: 32