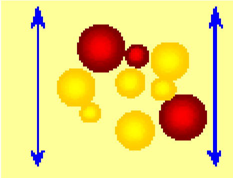
1 Set 1

Use Greedy method

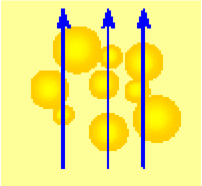
bibliography:
FRENTIU M., POP H.F., SERBAN G., Programming Fundamentals, 2006

1.1 Problems

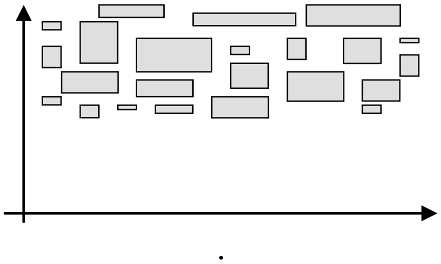
1. Consider *n* balloons that are moving vertically. Select the biggest number of balloons such that they will not touch between them. *(Present the list of balloons of the solution.)*

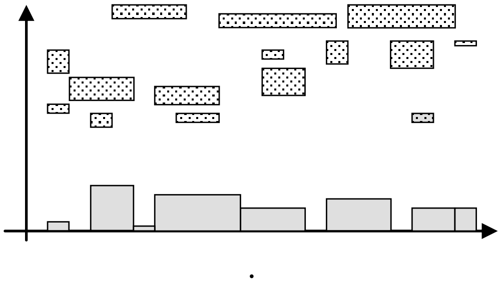


2. Given n balloons, determine the smallest number of arrows to break all the balloons. It is considered that the arrows are launched vertically. *(Present the arrow positions of the solution)*



3. A rain of meteorites of rectangular shape (with the sides parallel with the axes) are falling on Earth (the *Ox* axis). A part of these meteorites must be destroyed to avoid overlapping when touching the ground. Which meteorites must be destroyed such that the maximum number will remain on ground?





Hints:
Balloons are circles considered in plane and can move vertically in both directions.
Meteorites are of rectangular shape in a vertical plane.

1.2 Themes

Choose one of the problems from the list above (see section problem) and solve it by using the given ADT(s) and two DS to implement it.

No	ADT	Representation 1	Representation 2
1.	List Iterator, bidir, read-only	Dynamic vector	doubly linked list
2.	List Iterator, FWD, read-only	dynamic vector	singly linked list
3.	Sorted List Iterator, FWD	dynamic vector	singly linked list
4.	Sorted List Iterator, bidirectional	dynamic vector	doubly linked list
5.	Sorted List	dynamic vector	binary search tree
6.	Sorted List	dynamic vector	red black tree
7.	Sorted List	linked list	binary search tree
8.	Sorted Set	dynamic vector	binary search tree
9.	Sorted Set	dynamic vector	red black tree
10.	Priority Queue	linked list	binary search tree
11.	Priority Queue	linked list	heap
12.	Priority Queue	linked list	red black tree

2 Set 2

Consider the next general problem:
find all words in a dictionary that can be formed with the letters of a given word.
E.g.: orchestra / carthorse
steak/skate

2.1 Problems:

1. Find all the anagrams of a given word
An anagram of a word is the result of rearranging the letters of a word to produce a new word, using all the original letters, and having the same letter frequencies.
2. Find all words in a dictionary that can be formed with the letters of a given word:
use the same letters, all the letters, but letters can have different frequencies
3. Find all words in a dictionary that can be formed with letters of a given word:
use the same letters, but not necessarily all of them; letters can have different frequencies

2.2 Themes

Choose one of the problems from the list (see section problem) and solve it by using the given ADT and two DS to implement it.
Use a list of at least 5000 words; consider only words longer more than 3 characters long.

No	ADT	Representation 1	Representation 2
13.	Sorted Map *or MultiMap*	dynamic vector	binary search tree
14.	Sorted Map *or MultiMap*	dynamic vector	red black tree
15.	Sorted Map *or MultiMap*	linked list	binary search tree
16.	Sorted Map *or MultiMap*	linked list	red black tree
17.	Map *or MultiMap*	dynamic vector	hashtable, collision resolution through chaining; use vector
18.	Map *or MultiMap*	dynamic vector	hashtable, collision resolution through chaining; use sorted singly linked list
19.	Map *or MultiMap*	dynamic vector	hashtable, collision resolution through chaining; use sorted doubly linked list
20.	Map *or MultiMap*	dynamic vector	hash table, collision resolution through open addressing
21.	Set	dynamic vector	hashtable, collision resolution through chaining; use sorted singly linked list
22.	Set	dynamic vector	hashtable, collision resolution through chaining; use binary search tree
23.	Set	dynamic vector	hash table, collision resolution through open addressing

Remark:
You can find English word list on the Internet. You can try here:
http://wordlist.sourceforge.net/

3 Set 3

It is required to use breadth first search.
(***No other solutions will be considered.***)

- you can also study branch and bound method
FRENTIU M., POP H.F., SERBAN G., Programming Fundamentals, 2006)

3.1 Problem:

Path in a maze (labyrinth)
A robot is asked to navigate a maze.
It is placed at a certain position (the *starting* position: S) in the maze and is asked to try to reach another position (the *goal* position: G).
Maze has rectangular shape. Positions are identified by (x,y) coordinates.
Positions in the maze will either be open (*) or blocked with an obstacle (X).
At any given moment, the robot can only move 1 step in one of 4 directions. Valid moves are:

- Go North: (x,y)
- Go East: (x,y)
- Go South: (x,y)
- Go West: (x,y)

The robot can only move to positions without obstacles and must stay within the maze.
The robot should search for a path from the starting position (S) to the goal position (G) until it finds one or until it exhausts all possibilities.

S	*	X	X	*	*	*
*	X	*	*	X	*	*
*	*	*	*	*	*	*
*	X	*	X	*	*	X
*	X	*	*	*	*	X
*	*	*	*	X	*	G
*	X	*	X	*	*	*

Find optimum solution (shortest path).

3.2 Themes

Choose one of the problems from the list (see section problem) and solve it by using the given ADT(s) and two DS to implement it.

No	ADT	Representation 1	Representation 2
24.	Stack Queue	dynamic vector	singly linked list
25.	Stack Queue	dynamic vector	doubly linked list
26.	Deque	dynamic vector	doubly linked list
27.	Deque	singly linked list	doubly linked list
28.	Deque	dynamic vector	singly linked list
29.	List Iterator, bidir., read-only	dynamic vector	doubly linked list
30.	List with indexed access	dynamic vector	singly linked list
31.	List with indexed access	dynamic vector	doubly linked list
32.	List with position based access	dynamic vector	linked list
33.	List Iterator, FWD, RW	dynamic vector	singly linked list
34.	List Iterator, bidir., RW	dynamic vector	doubly linked list