**package** dougTest;

**import** java.util.\*;

**public** **class** Employee\_Comparable\_and\_Iterator **implements** Comparable<Object> {

**int** EmpID;

String Ename;

**double** Sal;

**static** **int** *i*;

**public** Employee\_Comparable\_and\_Iterator() {

EmpID = *i*++;

Ename = "dont know";

Sal = 0.0;

}

**public** Employee\_Comparable\_and\_Iterator(String ename, **double** sal) {

EmpID = *i*++;

Ename = ename;

Sal = sal;

}

**public** String toString() {

**return** "EmpID " + EmpID + "\n" + "Ename " + Ename + "\n" + "Sal " + Sal;

}

**public** **int** compareTo(Object o1) {

**if** (**this**.Sal == ((Employee\_Comparable\_and\_Iterator) o1).Sal)

**return** 0;

**else** **if** ((**this**.Sal) > ((Employee\_Comparable\_and\_Iterator) o1).Sal)

**return** 1;

**else**

**return** -1;

}

**public** **static** **void** main(String[] args) {

List<Employee\_Comparable\_and\_Iterator> ts1 = **new** ArrayList<Employee\_Comparable\_and\_Iterator>();

ts1.add(**new** Employee\_Comparable\_and\_Iterator ("Tom",40000.00));

ts1.add(**new** Employee\_Comparable\_and\_Iterator ("Harry",20000.00));

ts1.add(**new** Employee\_Comparable\_and\_Iterator ("Maggie",50000.00));

ts1.add(**new** Employee\_Comparable\_and\_Iterator ("Chris",70000.00));

Collections.*sort*(ts1);

Iterator<Employee\_Comparable\_and\_Iterator> itr = ts1.iterator();

**while**(itr.hasNext()){

Object element = itr.next();

System.***out***.println(element + "\n");

}

}

}

**import** java.util.\*;

**public** **class** Iter<T> **implements** Iterable<T> {

**private** T[] a = (T[]) **new** Object[2];

**private** **int** N = 0;

**public** **boolean** isEmpty() {

**return** N == 0;

}

**public** **int** size() {

**return** N;

}

**private** **void** resize(**int** max) {

T[] temp = (T[]) **new** Object[max];

**for** (**int** i = 0; i < N; i++)

temp[i] = a[i];

a = temp;

}

**public** **void** push(T item) {

**if** (N == a.length)

resize(2 \* a.length);

a[N++] = item;

}

**public** T pop() {

T item = a[--N];

// a[N] = null;

**if** (N > 0 && N == a.length / 4)

resize(a.length / 2);

**return** item;

}

**public** Iterator<T> iterator() {

**return** **new** ReverseArrayIterator();

}

**private** **class** ReverseArrayIterator **implements** Iterator<T> {

**private** **int** i = N;

**public** **boolean** hasNext() {

**return** i > 0;

}

**public** T next() {

**return** a[--i];

}

**public** **void** remove() {

}

}

**public** **static** **void** main(String[] args) {

ResizingArrayStack<Integer> s = **new** ResizingArrayStack();

**for** (**int** i = 1; i < 9; i++)

s.push(i);

**for** (**int** x : s)

System.***out***.print(x + " ");

System.***out***.println();

// while (!StdIn.isEmpty()) {

// int item = Integer.parseInt(StdIn.readString());

// if (item != 0) s.push(item);

// else if (!s.isEmpty()) StdOut.print(s.pop() + " ");

// }

// StdOut.println("(" + s.size() + " left on stack)");

**int**[] a = { 1, 2, 3, 4 };

**int**[] b = { 5, 6, 7, 8, 9 };

**for** (**int** x : a)

System.***out***.print(x);

System.***out***.println();

**for** (**int** x : b)

System.***out***.print(x);

System.***out***.println();

ArrayList<String> al = **new** ArrayList<String>();

// add elements to the array list

al.add("C");

al.add("A");

al.add("E");

al.add("B");

al.add("D");

al.add("F");

**for** (String st : al)

System.***out***.print(st + " ");

System.***out***.println();

// Use iterator to display contents of al

System.***out***.print("Original contents of al: ");

Iterator<String> itr = al.iterator();

**while** (itr.hasNext()) {

Object element = itr.next();

System.***out***.print(element + " ");

}

System.***out***.println();

// Modify objects being iterated

ListIterator<String> litr = al.listIterator();

**while** (litr.hasNext()) {

Object element = litr.next();

litr.set(element + "+");

}

System.***out***.print("Modified contents of al: ");

itr = al.iterator();

**while** (itr.hasNext()) {

Object element = itr.next();

System.***out***.print(element + " ");

}

System.***out***.println();

// Now, display the list backwards

System.***out***.print("Modified list backwards: ");

**while** (litr.hasPrevious()) {

Object element = litr.previous();

System.***out***.print(element + " ");

}

System.***out***.println();

}

}

**package** dougTest;

**import** java.util.LinkedList;

**public** **class** Queue2 {

**private** LinkedList<Integer> data = **new** LinkedList<Integer>();

**public** **void** enqueue(Integer item) {data.addLast(item); }

**public** Integer dequeue() {**return** data.removeFirst(); }

**public** Integer peek() {**return** data.getFirst(); }

**public** **int** size() {**return** data.size(); }

**public** **boolean** isEmpty() {**return** data.isEmpty(); }

**public** **static** **void** main(String[] args)

{

//Build & Print queue from customized class elements

Queue2 q = **new** Queue2();

**for** (**int** i = 1; i < 5; i++)

q.enqueue(i);

**while** (!q.isEmpty())

System.***out***.println(q.dequeue());

System.***out***.println("----");

//Build & Print queue from Java elements

LinkedList<Integer> queue = **new** LinkedList<Integer>();

**for** (**int** i = 1; i < 5; i++)

queue.add(i); //same as addLast(i)

**while** (!queue.isEmpty())

System.***out***.println(queue.removeFirst());

}

}

**package** dougTest;

**import** java.util.NoSuchElementException;

**public** **class** LinkedStackOfStrings {

**private** Node first = **null**;

**private** **class** Node {

String item;

Node next;

}

**public** **boolean** isEmpty() {

**return** first == **null**;

}

**public** **void** push(String item) {

Node oldfirst = first;

first = **new** Node();

first.item = item;

first.next = oldfirst;

}

**public** String pop() {

**if** (isEmpty())

**throw** **new** NoSuchElementException("Stack underflow");

String item = first.item;

first = first.next;

**return** item;

}

**public** **static** **void** main(String[] args) {

LinkedStackOfStrings ls = **new** LinkedStackOfStrings();

**for** (**int** i = 1; i < 10; i++) {

String s = i + "";

System.***out***.print(s);

ls.push(s);

}

System.***out***.println();

**while** (!ls.isEmpty())

System.***out***.print(ls.pop() + " ");

}

}

**package** dougTest;

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\* A program with N log N running time. Read in N integers

\* and counts the number of pairs that sum to exactly 0.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**import** java.util.Arrays;

**public** **class** TwoSumFast {

// print distinct pairs (i, j) such that a[i] + a[j] = 0

**public** **static** **void** printAll(**int**[] a) {

**int** N = a.length;

Arrays.*sort*(a);

**for** (**int** i = 0; i < N; i++) {

**int** j = Arrays.*binarySearch*(a, -a[i]);

**if** (j > i) StdOut.*println*(a[i] + " " + a[j]);

}

}

// return number of distinct pairs (i, j) such that a[i] + a[j] = 0

**public** **static** **int** count(**int**[] a) {

**int** N = a.length;

Arrays.*sort*(a);

**int** cnt = 0;

**for** (**int** i = 0; i < N; i++) {

**int** j = Arrays.*binarySearch*(a, -a[i]);

**if** (j > i) cnt++;

}

**return** cnt;

}

**public** **static** **void** main(String[] args) {

**int**[] a = {-405375, 1, 2, 405375, -192459, 3,4, 192459, -77867, 5, 77867};

**int** cnt = *count*(a);

StdOut.*println*(cnt);

*printAll*(a);

}

}

------------------------

**public** **class** Comp\_Lambda\_SRM166\_Div2\_L2\_BinaryCardinality

{

**public** **int**[] arrange(**int**[] numbers)

{

**int** n = numbers.length;

Integer[] nums = **new** Integer[n];

**for** (**int** i = 0; i < n; i++) nums[i] = numbers[i];

Arrays.*sort*(nums, (o1,o2) -> (Integer.*bitCount*(o1) - Integer.*bitCount*(o2) != 0) ?

Integer.*bitCount*(o1) - Integer.*bitCount*(o2) : o1 - o2 );

**for** (**int** i = 0; i < n; i++) numbers[i] = nums[i];

**return** numbers;

}

**public** **class** Comp\_R280\_Div2\_C\_Alt2

{

FastReader in;

PrintWriter out;

**public** **static** **void** main(String[] args) {

**new** Comp\_R280\_Div2\_C\_Alt2().run();

}

**void** run()

{

in = **new** FastReader(System.***in***);

out = **new** PrintWriter(System.***out***);

solve();

out.close();

}

**class** Pair //implements Comparable<Pair> //Option 1

{

**int** a, b;

**public** Pair(**int** a, **int** b) {

**this**.a = a;

**this**.b = b;

}

//@Override

//public int compareTo(Pair o) { //Option 1

// return this.b - o.b;

// return Integer.compare(b, o.b); //Option 1b

//}

}

//Option 4

//static class IntCompare implements Comparator<Pair> { //Option 4

// public int compare(Pair o1, Pair o2) {

// return o1.b - o2.b;

// }

//}

**void** solve()

{

**int** n = in.nextInt();

**int** r = in.nextInt();

**long** avg = in.nextInt();

**int**[] a = **new** **int**[n];

**int**[] b = **new** **int**[n];

**long** sum = 0;

Pair[] c = **new** Pair[n];

**for** (**int** i = 0; i < n; i++)

{

a[i] = in.nextInt();

b[i] = in.nextInt();

c[i] = **new** Pair(a[i], b[i] );

sum += a[i];

}

//Arrays.sort(c); //Option 1

//Option 2

//Arrays.sort(c, (o1, o2) -> o1.b.compareTo(o2.b)); //b must be Integer not int

Arrays.*sort*(c, (o1, o2) -> o1.b - o2.b); //Opt 2b

//Arrays.sort(c, (o1, o2) -> Integer.compare(o1.b, o2.b)); //Opt 2c

//Option 3

//Arrays.sort(c, new Comparator<Pair>() {

// @Override //Optional

// public int compare(Pair o1, Pair o2) {

// return o1.b - o2.b;

// }

//});

//Option 4

//Comparator<Pair>comp = new IntCompare();

//Arrays.sort(c, comp);

**long** sumNeeded = avg \* n;

**long** dif = sumNeeded - sum;

**long** essays = 0;

**int** i = 0;

**while** (sumNeeded > sum && i < n)

{

dif = sumNeeded - sum;

**long** min = Math.*min*(dif, r - c[i].a);

essays += (min \* c[i].b);

sum += min;

i++;

}

out.println(essays);

}

# Java Iterables

SequentialSearchST.java in Princeton

/\*\*

\* Returns all keys in the symbol table as an <tt>Iterable</tt>.

\* To iterate over all of the keys in the symbol table named <tt>st</tt>,

\* use the foreach notation: <tt>for (Key key : st.keys())</tt>.

\* **@return** all keys in the sybol table as an <tt>Iterable</tt>

\*/

**public** Iterable<Key> keys() {

Queue<Key> queue = **new** Queue<Key>();

**for** (Node x = first; x != **null**; x = x.next)

queue.enqueue(x.key);

**return** queue;

}

/\*\*

\* Unit tests the <tt>SequentialSearchST</tt> data type.

\*/

**public** **static** **void** main(String[] args) {

SequentialSearchST<String, Integer> st = **new** SequentialSearchST<String, Integer>();

String[] st1 = **new** String[] {"Doug", "Craig", "Brad", "Egor"};

**for** (**int** i = 0; i < 4; i++) {

String key = st1[i];

st.put(key, i);

}

**for** (String s : st.keys())

System.***out***.println(s + " " + st.get(s));

}