

Forum:[Hack Forums](#)

Thread: [Issues With Merge Sort Function](#)

S?e?f - February 22, 2020, 5:22 pm

So for a class assignment we have to merge sort a vector of custom classes called Record. I think I'm having an issue when it comes to merging back together the split L/R sections.

Heres my driver code:

```
Code void CensusData::mergeSort(int /*type*/)
```

```
{
    mergeSortPop(data, 0, data.size() - 1);
}And heres the rest of the merge sort algo:
```

```
Code void CensusData::mergeSortPop(vector<Record*>, int p, int r)
```

```
{
    if(p < r)
    {
        int q = (p+(r-1))/2;

        mergeSortPop(data, p, q);
        mergeSortPop(data, q + 1, r);
```

```
        mergePop(data, p, q, r);
    }
}
```

```
void CensusData::mergePop(vector<Record*> data, int p, int q, int r)
```

```
{
    int i, j, k;
    int n1 = q - p + 1;
    int n2 = r - q;

    vector<Record*> left;
    vector<Record*> right;
    for(i = 0; i < n1; i++)
    {
        left.push_back(data[p + i]);
        std::cout << "Pushing back" << *(data[p+i]) << "\n";
    }
    for(j = 0; j < n2; j++)
    {
        right.push_back(data[q + 1 + j]);
    }
    i = 0; j = 0; k = p;
    while(i < n1 && j < n2)
    {
        if(left[i]->population <= right[j]->population)
        {
            *(data[k]) = *(left[i]);
            i++;
        }else
        {

```

```

        *(data[k]) = *(right[j]);
        j++;
    }
    k++;
}
while(i < n1)
{
    (data[k]) = (left[i]);
    i++;
    k++;
}
while(j < n2)
{
    (data[k]) = (right[j]);
    j++;
    k++;
}
}Record holds pointers to population for each record created. Thanks guys.

```

SIM-3991524 - February 22, 2020, 10:53 pm

Maybe you could post more of the code?

Working mergesort with your Record class

Spoiler ([Click to View](#))

Code #include <iostream>

#include <cstdlib>

#include <vector>

#include <iomanip>

#define N_RECORDS 100

#define MAX_POPULATION 1000000

using namespace std;

class Record {

public:

int population;

Record(int population) {

this->population = population;

}

};

vector<Record> merge(vector<Record> a, vector<Record> b) {

int a_index = 0;

int b_index = 0;

vector<Record> out;

while (a_index < a.size() && b_index < b.size()) {

if (a[a_index].population < b[b_index].population) {

out.push_back(a[a_index]);

a_index++;

}

else {

```

        out.push_back(b[b_index]);
        b_index++;
    }
}

while (a_index < a.size()) {
    out.push_back(a[a_index]);
    a_index++;
}
while (b_index < b.size()) {
    out.push_back(b[b_index]);
    b_index++;
}

return out;
}

vector<Record> mergeSort(vector<Record> v) {
    if (v.size() < 2)
        return v;

    vector<Record> left(v.begin(), v.begin() + (v.size() / 2));
    vector<Record> right(v.begin() + (v.size() / 2), v.end());

    left = mergeSort(left);
    right = mergeSort(right);

    return merge(left, right);
}

void PrintRecords(vector<Record> records) {
    int i = 0;
    for (Record r : records) {
        if (i++ % 10 == 0)
            cout << endl << setw(5) << (i - 1);
        cout << setw(8) << r.population;
    }
}

int main()
{
    vector<Record> records;
    int pop;

    for (int i = 0; i < N_RECORDS; i++) {
        records.push_back(Record((rand() * rand()) % (MAX_POPULATION + 1)));
    }
    cout << "Random:" << endl;
    PrintRecords(records);
    records = mergeSort(records);
    cout << endl << endl << "Sorted:";
    PrintRecords(records);
}

```

```
cin >> pop;  
}
```

Example output

Spoiler ([Click to View](#))

Code Random:

```
0 757147 850833 413055 970788 597709 567065 748996 890847 766255 239546  
10 37691 597006 615541 51110 289218 341404 427628 215491 15758 544122  
20 446450 89276 426790 752135 41410 220562 908065 655526 970659 417541  
30 379023 873940 670520 835600 290636 351377 923704 79126 701140 546278  
40 776620 571656 70818 629180 496382 645867 839920 255787 557643 268210  
50 378645 583325 450495 230930 31069 328501 494015 547794 997805 161580  
60 773622 724226 665732 402602 581564 452469 25916 903333 150498 145155  
70 919970 159111 144872 295108 219498 203052 402822 733906 331871 51079  
80 710080 313521 695545 511284 708613 332959 301377 400406 445618 550831  
90 202215 105947 65346 137174 465301 521049 160430 110704 689176 42665
```

Sorted:

```
0 15758 25916 31069 37691 41410 42665 51079 51110 65346 70818  
10 79126 89276 105947 110704 137174 144872 145155 150498 159111 160430  
20 161580 202215 203052 215491 219498 220562 230930 239546 255787 268210  
30 289218 290636 295108 301377 313521 328501 331871 332959 341404 351377  
40 378645 379023 400406 402602 402822 413055 417541 426790 427628 445618  
50 446450 450495 452469 465301 494015 496382 511284 521049 544122 546278  
60 547794 550831 557643 567065 571656 581564 583325 597006 597709 615541  
70 629180 645867 655526 665732 670520 689176 695545 701140 708613 710080  
80 724226 733906 748996 752135 757147 766255 773622 776620 835600 839920  
90 850833 873940 890847 903333 908065 919970 923704 970659 970788 997805
```