Question 16.10 from CTCI: Living People

Benjamin De Brasi

January 24, 2018

Question

Living People: Given a list of people with their birth and death years, implement a method to compute the year with the most number of people alive. You may assume that all people were born between 1900 and 2000 (inclusive). If a person was alive during any portion of that year, they should be included in that year 's count. For example, Person (birth = 1908, death = 1909) is included in the counts for both 1908 and 1909.

Explanation and Algorithm

Here are three ways to solve this problem:

The first step in for the brute force solution is to create an array the size of the span of years (101 in this case). Each index in that array will correspond a year (so index 0 is 1900, index 1 is 1901, etc). Then traverse through the array with all the people objects looking at both their birth and death date. Add 1 to each array index corresponding to the to each year they were born stopping at the index of the death date + 1. Once you've populated the years array you will have the total amount of people alive in each year. Simply traverse the array keeping track of the max and the index where the max appears. Transform the index to a year and return it.

A second way to solve this problem is to create 2 arrays with all the people objects. One will be sorted by birth year and the other will be sorted by death year. Traverse through both arrays adding to a variable storing the amount of people currently alive while comparing it to the max amount of people you've seen alive at a given point. If the current is greater than the max, update max and the index. While doing all of this check the death list and every time you encounter someone with a death year subtract it from the currently alive count. Make sure you subtract the death years AFTER counting the birth years that way you avoid not counting someone who, for example, was born and died the same year.

The final way avoids sorting all together. Create an array with each index corresponding to each year we're analyzing. Then, traverse through the array with people objects adding a +1 to the year array to the matching index for every birth and a -1 to the year array for every death to the matching index +1. The reason it's matching index +1 instead of just matching index is to deal with the fact that people are obviously still alive at some point in the year they die. Once you've gone through all the people, your year array will contain the net change in populations from year to year. You can traverse through that array from start to end keeping track of the sum as you go. Whichever index corresponds to the sum being the largest is the year when the most people were alive.

Hints

1. Solution 1: Can you count the number of people alive in each year?
2. Solution 1: Try using a hash table, or an array that maps from a birth year to how many people are alive in that year.
3. Solution 2: What if you sorted the years? What would you sort by?
4. Solution 2: Do you actually need to match the birth years and death years? Does it matter when a specific person died, or do you just need a list of the years of deaths?
5. Solution 2: Observe that people are"fungible:' lt doesn't matter who was born and when they died. All you need is a list of birth years and death years. This might make the question of how you sort the list of people easier.
6. Solution 2: Try creating a sorted list of births and a sorted list of deaths. Can you iterate through both, tracking the number of people alive at anyone time?
7. Solution 3: Each birth adds one person and each death removes a person. Try writing an example of a list of people (with birth and death years) and then re-formatting this into a list of each year and a + 1 for a birth and a -1 for a death.
8. Solution 3: What if you created an array of years and how the population changed in each year? Could you then find the year with the highest population?
9. Solution 3: Be careful with the little details in this problem. Does your algorithm/code handle a person who dies in the same year that they are born? This person should be counted as one person in the population count.

Code

```
/*Answer 1 */

int maxAliveYear(Person[] people, int min, int max){
   int[] years = createYearMap(people, min, max);
   int best = getMaxIndex(years);
   return best + min;
}

/* Add each person's years to a year map. */
int[] createYearMap(Person[] people, int min, int max){
   int[] years = new int[max - min + 1];
   for (Person person : people) {
      incrementRange(years, person. birth - min, person.
   }
   return years;
}

/ * Increment array for each value between left and right */
void incrementRange(int[] values, int left, int right){
   for (int i = left; i (= right; i++) {
      values[i]++j
   }
}

/* Get index of largest element in array. */
int get Maxlndex(int[] values){
   int max = 0;
   for (int i = lj i < values.lengthj i++) {
      if (values[i] > values[max]) {
         max = i;
      }
   }
   return max;
}
}
```

```
/* Answer 2 */

int maxAliveYear(Person[] people, int min, int max) {
   int[] births getSortedYears(people, true);
   int[] deaths = getSortedYears(people, false);
   int birthlndex = 0;
   int deathlndex = 0;
   int currentlyAlive 0;
```

```
  int maxAlive = 0;
  int maxAliveYear = min;

  /* Walk through arrays. */
  while (birthlndex < births. length) {
    if (births[birthlndex] <= deaths[deathlndex]) {
      currentlyAlive++; II include birth
      if(currentlyAlive > maxAlive) {
        maxAlive = currentlyAlive;
        maxAliveYear = births[birthlndex];
      }
      birthlndex++; II move birth index
    } else if (births[birthlndex] > deaths[deathlndex]) {
      currentlyAlive--; // include death
      deathlndex++; // move death index
    }
  }
  return maxAliveYear;
}
/* Copy birth years or death years (depending on the value of
    copyBirthYear into int array and sort it */

int[] getSortedYears(Person[] people, boolean copyBirthYear){
  int[] years = new int[people . length];
  for (int i = 0; i < people. length; i++) {
    years[i] = copyBirthYear ? people[i] . birth people[i].death;
  }
  Arrays.sort(years);
  return years;
}
```

```
/* Answer 3 */

int maxAliveYear(Person[] people, int min, int max) {
  /* Build population delta array. */
  int[] populationDeltas = getPopulationDeltas(people, min, max);
  int maxAliveYear = getMaxAliveYear(populationDeltas);
  return maxAliveYear + min;
}

/* Add birth and death years to deltas array. */
int[] getPopulationDeltas(Person[] people, int min, int max) {
  int[] populationDeltas = new int[max - min + 2];
  for (Person person : people) {
    int birth = person.birth - min;
    populationDeltas[birth]++;

    int death = person.death - min;
    populationDeltas[death + 1]--;
  }
  return populationDeltas;
}

/* Compute running sums and return index with max. */
int getMaxAliveYear(int[] deltas) {
  int maxAliveYear = 13;
  int maxAlive = 13;
  int currentlyAlive = e;
  for (int year = 13; year < deltas. length; year++) {
    currentlyAlive += deltas[year];
    if (currentlyAlive > maxAlive) {
      maxAliveYear = year;
      maxAlive = currentlyAlive;
    }
  }
  return maxAliveYear;
}
```

Big O analysis

Below are the Big O analyses for each solution. For each, P = num of people, Y = years of each person's lifespan and R = the range of years the problem is covering:

1. The brute force solution has a run time of (PY +R) where PY comes from having to traverse and fill the years array with +1s for each year each person is alive and R comes from having to traverse the years array once it's been populated to find the year with the most people currently alive. Since in the worst case, Y = R this is in O(n^2) algorithm.

2. The algorithm involving sorting is O(P log P) which comes from the sort itself. This is the most time consuming part of the algorithm because once you've sorted it's just 2 O(P) operations to traverse through the sorted death years and sorted birth years array while adding and subtracting a currentlyAlive variable and changing a maxAlive and an index variable for where the maxAlive value was found.

3. The final algorithm is O(R + P). The P comes from traversing through all the people while adding and subtracting 1 to corresponding indices in the years array and R comes form traversing through the years array to find the index with the most people alive.

Sources

Question, answer and other material taken from Cracking the Coding Interview 6th edition by Gayle Laakmann McDowell.