Algorithms and Data Structures

Second practical assignment

Rowan Goemans ($s4375238$) and Ties Klappe ($s1030293$)

January 8, 2019

1 Explanation

1.1 The problem

The problem states that the students of the course Algorithms and Data Structures in the year 2021 want to figure out the correct answers to the latest (multiple-choice) exam. The information they have, for each student, are the answers that he/she gave for all the questions as well as the total number of correct answers. If there is no solution or more than one, the algorithm must compute the total number of solutions that is consistent with the information the students have.

1.2 The algorithm

Our algorithm reads a $n \times m$ matrix A existing out of $1 \leq n \leq 12$ students and $1 \leq m \leq 40$ questions. First, we split m in two halves ($\lfloor \frac{m}{2} \rfloor$ and $\lceil \frac{m}{2} \rceil$) $m = m_1 + m_2$, and divide A into a $n \times m_1$ matrix A_1 and a $n \times m_2$ matrix A_2. For A_1 we generate all the 2^{m_1} possible solutions X_1, and for A_2 we generate all the 2^{m_2} possible solutions X_2. Now, we add the vector $x_1 \in X_1$ to the possible solutions P_1 of A_1 if for every student n it holds that:

$$A_1[n] \odot x_1 \leq r(A_1[n]) \leq s(n)$$

Here we have defined the following operations:

- $A_1[n] \odot x_1$: the number of correct answers given to each question $q \in m_1$ by student n using response model x_1
- $r(A_1[n])$: the minimum number of answers student n must have answered correct of the questions m_1, calculated by $s(n) - |m_2|$
- $s(n)$: the total score of student n

Similarly, we add the vector $x_2 \in X_2$ to the possible solutions P_2 of A_2 if for every student n it holds that $A_2[n] \odot x_2 \leq r(A_2[n]) \leq s(n)$. Here we have defined the following operations:

- $A_2[n] \odot x_2$: the number of correct answers given to each question $q \in m_2$ by student n using response model x_2
- $r(A_2[n])$: the minimum number of answers student n must have answered correct of the questions m_2, calculated by $s(n) - |m_1|$

Now that we have two lists of vectors (P_1 and P_2) that contain the possible response models for A_1 and A_2, we can start combining the partial solution in P_1 and P_2. For each $p_j \in P_2$ we look for matching solutions $p_i \in P_1$ such that for each student n the sum of the partial scores based on p_i and p_j for n equal $s(n)$. Finally, we output the number of solutions or print the solution if there is precisely one solution.

1.3 Data structures

The possible solutions P_1 and P_2, mentioned in the previous paragraph, are stored in a hash table (C++ *unordered map*), where the scores of each student are used as the key and the associated solution(s) as the value. The hashes are generated by a perfect hashing function: since we split each exam into two, each student has a maximum partial score of 20 (since $1 \leq \frac{m}{2} \leq 20$). This score can be stored in 5 bits (since $2^5 = 32$). Now, the maximum number of students is 12, so the information of one set of scores for each student can be stored in $12 \cdot 5 = 60$ bits. So this shows that the hash for a set of scores can be stored in an uint_64 C++ datatype. In the case that a generated set of scores already exists, we add the response model to the list of response models that already exist with that set of scores. This assures we can find matching solutions $p_i \in P_1$ (if they exist) for every $p_j \in P_2$ in $\Theta(m)$. To store the set of scores for a response model, we use C++'s valarray. When we have the possible response models P_1 and P_2, we loop through each $p_j \in P_2$. The valarray allows us to, without having to explicitly check the score of each student, find $p_i \in P_1$ such that for every student n, the partial score of n of p_i plus the partial score of n of p_j equals $s(n)$. So instead of $\mathcal{O}(n \times m)$ we have reduced the complexity to $\mathcal{O}(m)$.

The $\odot$ operation we defined in the previous paragraph is implemented by two binary operations. Given a partial solution of a student s and a response model x, we first XOR the partial solution with x. This results in a bitstring where we have a 0 at position i if question i was answered correctly by s according to x and a 1 otherwise. The second operation is negating the bit string, flipping all $1's$ and $0's$ resulting in a bit string with a 1 at every position i where i is a correctly answered question. This allows us to use *__builtin_popcount* (to count student s his score) that counts the number of flipped bits with a single Assembly instruction.

2 Correctness

We show correctness by proving our algorithm is equivalent to the complete bruteforce appreach where all 2^n bit sequences are tested and a solution is only valid if $\forall i \in \{1, 2, \cdots, n\}\ [A[i] \odot x = s(i)]$.

When splitting the answer into two parts we know the following must hold for our algorithm to be correct:

$$S := \{1, 2, \cdots, n\}$$
$$\forall i \in S\ [A_1[i] \odot x_1 + A_2[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A[i] \odot x = s(i)]$$

We can prove this using induction.

Theorem 1. *let the predicate P be defined by:*

$$P((p,q)) := \forall i \in S\ [A_1^p[i] \odot x_1 + A_2^q[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A^{p+q}[i] \odot x = s(i)]$$

Then $P((p,q))$ holds for all (p,q) such that $(p+q) \geq 1$.

Proof by induction on (p,q).

Basis step $P((p,q))$ holds when $p+q=1$. We make a case distinction between when $p=1, q=0$ and $p=0, q=1$.

Case $p=0, q=1$:

$$\begin{aligned}
&\forall i \in S\ [A_1^0[i] \odot x_1 + A_2^1[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A^1[i] \odot x = s(i)] \\
=&\forall i \in S\ [0 + A^1[i] \odot x = s(i)] \iff \forall i \in S\ [A^1[i] \odot x = s(i)] \\
=&\forall i \in S\ [A^1[i] \odot x = s(i)] \iff \forall i \in S\ [A^1[i] \odot x = s(i)]
\end{aligned}$$

. Case $p=1, q=0$:

$$\begin{aligned}
&\forall i \in S\ [A_1^1[i] \odot x_1 + A_2^0[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A^1[i] \odot x = s(i)] \\
=&\forall i \in S\ [A[i] \odot x + 0 = s(i) + 0] \iff \forall i \in S\ [A^1[i] \odot x = s(i)] \\
=&\forall i \in S\ [A[i] \odot x = s(i)] \iff \forall i \in S\ [A^1[i] \odot x = s(i)]
\end{aligned}$$

Inductive step

Let (k,r) with $k+r \geq 1$ such that $P((k,r))$ holds.

- Hence $P((k,r)) := \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A^{k+r}[i] \odot x = s(i)]$ holds. (IH).

- We have to prove that $P((k+1,r))$ holds and $P((k,r+1))$.
- Hence we have to prove that

  $$P((k+1,r)) = \forall i \in S\ [A_1^{k+1}[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A^{k+1+r}[i] \odot x = s(i)]$$

  holds, and that

  $$P((k,r+1)) = \forall i \in S\ [A_1^{k}[i] \odot x_1 + A_2^{r+1}[i] \odot x_2 = s(i)] \iff \forall i \in S\ [A^{k+r+1}[i] \odot x = s(i)]$$

  holds.
- This holds because we can again make a case distinction between when the new answer results in the student getting an extra point and the student not getting an extra point.

  Case $P((k+1,r))$ and student gets an extra point:

  For every student it holds that when making the exam a single question longer and this results in that student having a higher score then $s(i)$ is increased by one. Since we added that correct answer to A_1^{k+1} this means $A_1^{k+1} \odot x_1 = A_1^k \odot x_1 + 1$ and the score for the other part stays the same. However since the correct answer is also added to A^{k+r+1} this means, $A^{k+r+1} \odot x = A^{k+r} \odot x + 1$
  Thus:

  $$\begin{aligned} P((k+1,r)) &= \forall i \in S\ [A_1^{k+1}[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i) + 1] \\ &\iff \forall i \in S\ [A^{k+1+r}[i] \odot x = s(i) + 1] \\ &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 + 1 = s(i) + 1] \\ &\iff \forall i \in S\ [A^{k+r}[i] \odot x + 1 = s(i) + 1] \\ &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \\ &\iff \forall i \in S\ [A^{k+r}[i] \odot x = s(i)] \end{aligned}$$

  And by the induction hypothesis(IH) we know this holds.

  Case $P((k+1,r))$ and student doesn't get an extra point:

  For every student it holds that when making the exam a single question longer and this results in that student not getting a higher score then $s(i)$ stays the same. Since we added that wrong answer to A_1^{k+1} this means $A_1^{k+1} \odot x_1 = A_1^k \odot x_1$ and the score for the other part stays the same. However since the wrong answer was also added to A^{k+1+r} that means $A^{k+1+r} \odot x = A^{k+r} \odot x$.

Thus:

$$\begin{aligned} P((k+1,r)) &= \forall i \in S\ [A_1^{k+1}[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \\ &\iff \forall i \in S\ [A^{k+1+r}[i] \odot x = s(i)] \\ &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \\ &\iff \forall i \in S\ [A^{k+r}[i] \odot x = s(i)] \end{aligned}$$

And by the induction hypothesis(IH) we know this holds.

Case $P((k,r+1))$ and student gets an extra point:

For every student it holds that when making the exam a single question longer and this results in that student having a higher score then $s(i)$ is increased by one. Since we added that correct answer to A_1^{r+1} this means $A_1^{r+1} \odot x_1 = A_1^r \odot x_2 + 1$ and the score for the other part stays the same. However since the correct answer is also added to A^{k+1+r} this means, $A^{k+r+1} \odot x = A^{k+r} \odot x + 1$

Thus:

$$\begin{aligned} P((k,r+1)) &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^{r+1}[i] \odot x_2 = s(i)+1] \\ &\iff \forall i \in S\ [A^{k+r+1}[i] \odot x = s(i)+1] \\ &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 + 1 = s(i)+1] \\ &\iff \forall i \in S\ [A^{k+r}[i] \odot x + 1 = s(i)+1] \\ &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \\ &\iff \forall i \in S\ [A^{k+r}[i] \odot x = s(i)] \end{aligned}$$

And by the induction hypothesis(IH) we know this holds.

Case $P((k,r+1))$ and student doesn't get an extra point:

For every student it holds that when making the exam a single question longer and this results in that student not getting a higher score then $s(i)$ stays the same. Since we added that wrong answer to A_1^{r+1} this means $A_1^{r+1} \odot x_1 = A_1^r \odot x_1$ and the score for the other part stays the same. However since the wrong answer was also added to A^{k+r+1} that means $A^{k+r+1} \odot x = A^{k+r} \odot x$.

Thus:

$$\begin{aligned} P((k,r+1)) &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^{r+1}[i] \odot x_2 = s(i)] \\ &\iff \forall i \in S\ [A^{k+r+1}[i] \odot x = s(i)] \\ &= \forall i \in S\ [A_1^k[i] \odot x_1 + A_2^r[i] \odot x_2 = s(i)] \\ &\iff \forall i \in S\ [A^{k+r}[i] \odot x = s(i)] \end{aligned}$$

And by the induction hypothesis(IH) we know this holds.

This completes the Inductive step.

Now the principe of mathematical induction tells us that $P(p, q)$ holds for every $p, q \in \mathbb{N}$ where $p + q \geq 1$

3 Complexity

We will show our algorithm has a worst-case time complexity of $\mathcal{O}(n \cdot 2^{\lceil \frac{m}{2} \rceil})$, where m is the amount of questions and n is the amount of students. The algorithm runs through the following steps:

1) Reading input from standard in and putting it into arrays. This is linear in the amount of students $\mathcal{O}(n)$.

2) Calculating where to split the answers into two parts. This is $\mathcal{O}(1)$.

3) Computing the maps of scores for the first part. We loop over each binary sequence from 0 up to but not including $2^{\lceil \frac{m}{2} \rceil}$ and for each sequence we loop through all n students and check if that score is a score that could lead to a solution. If this is the case we add it to the map of solutions we have. We can check this in $\mathcal{O}(1)$ since this are just some bit operations. Adding to the map of solutions is $\mathcal{O}(n)$ since our hash function loops over the score of each student to create the hash. The total complexity of this is thus: $\mathcal{O}(2n \cdot 2^{\lceil \frac{m}{2} \rceil})$.

4) Similarly, we do this for the second part of the answer. This is identical to item 3), however this time we loop over binary sequences from 0 up to but not including $2^{\lfloor \frac{m}{2} \rfloor}$. Thus the complexity is: $\mathcal{O}(2n \cdot 2^{\lfloor \frac{m}{2} \rfloor})$.

5) We now start looping over every solution found for the second part and use the scores of that solution to find matching scores in the second part with associated solutions. Generating the key for the second map can be done in $\mathcal{O}(1)$ using the valarray data structure. A lookup of that key in the first map is $\mathcal{O}(n)$, since a lookup includes computing the hash. Combining the solution for the left part and right part into a complete solution is some bit fiddling and thus $\mathcal{O}(1)$. The total complexity of this operation is thus the number of solutions in the second part times n. In the worst case this is equal to $n \cdot 2^{\lfloor \frac{m}{2} \rfloor}$, if every student has exactly half the exam questions correct. Thus the complexity of this operation is $\mathcal{O}(n \cdot 2^{\lfloor \frac{m}{2} \rfloor})$.

6) We now print the result, which is $\mathcal{O}(1)$.

Thus the total complexity of the algorithm is:

$$\mathcal{O}(n) + \mathcal{O}(1) + \mathcal{O}(2n \cdot 2^{\lceil \frac{m}{2} \rceil}) + \mathcal{O}(2n \cdot 2^{\lfloor \frac{m}{2} \rfloor}) + \mathcal{O}(n \cdot 2^{\lfloor \frac{m}{2} \rfloor}) + \mathcal{O}(1)$$

The highest term dominates, so the complete algorithm has a complexity of:

$$\mathcal{O}(n \cdot 2^{\lceil \frac{m}{2} \rceil})$$