The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

cemc.uwaterloo.ca

2023 CCC Junior Problem Commentary

This commentary is meant to give a brief outline of what is required to solve each problem and what challenges may be involved. It can be used to guide anyone interested in trying to solve these problems, but is not intended to describe all the details of a correct solution. We encourage everyone to try and implement these details on their own using their programming language of choice.

J1 Deliv-e-droid

A small calculation involving the input values is needed to calculate the number of points gained based on the number of packages delivered and the number of points lost based on the number of collisions with obstacles. An if statement is also needed to possibly apply bonus points.

J2 Chili Peppers

This problem can be solved by using a variable to accumulate the total spiciness of Ron's chili. This value can be updated for each pepper name read from the input using a loop which iterates N times. One way to perform this update in the body of the loop is to use an if statement with a test for each possible pepper name. An alternative approach is to effectively place the table given in the problem statement in memory and then use the table to look up the SHU value given the name of a pepper. Different programming languages provide different ways of storing this table.

J3 Special Event

Each of the subtasks for this problem requires us to keep a count for each day of the number of people able to attend the event on that day. These can be kept in five separate variables because there are only five days. However, it is cleaner and probably less error-prone to store these counts in a list or array.

For the first subtask, we can simply search for a day with a count of N. We are told that there will only be one such day.

For the second subtask, we have to calculate the maximum of these five counts and then find the day corresponding to this maximum value. We are told there will only be one such day.

For the third and final subtask, we need to handle "ties". That is, there could be more than one day with the same maximum count value. It is a bit tricky to output the right answer in this case. It is important to avoid printing a comma after the final day number.

J4 Trianglane

To help Bocchi determine how much tape she needs, we need to do some careful counting.

For the first subtask, we can count B, the number of black triangles, which is the number of times 1 appears in the input. The correct answer will be $3 \times B$. Note that we can ignore the second line of input which is also true for the second subtask. However, for this second subtask, there could be adjacent black triangles and Bocchi does not need to place warning tape on the common side in these cases. Notice that each of these common sides contributes two to the value of $3 \times B$. This means we obtain the correct final answer by subtracting two from $3 \times B$ for each pair of adjacent black triangles.

The same principle applies to the third subtask but we also have to look for black triangles that are adjacent but in different rows. These can only occur at even-numbered positions (assuming we start counting at zero).

It is possible to solve this problem without making the observation that you can subtract from $3 \times B$. This approach involves looking for all the places where warning tape needs to be placed. An algorithm is needed which includes sides of triangles on the perimeter of the pathway as well as sides between adjacent triangles which are not the same colour.

Regardless of which approach is taken, many solutions to the first three subtasks will be quite efficient. However, the fourth subtask is meant to ensure this by disallowing overly slow solutions. It specifically requires that not too many passes are made over either row. More formally, only a constant number of passes of each row is allowed. Solutions that miss this requirement are probably too slow because they use a built-in function within a loop which itself loops through a row of triangles.

J5 CCC Word Hunt

The intended challenge of the final problem in this year's competition was different in nature than the last few years. Here, systematically considering each possible placement of the word in the grid will earn full marks if implemented correctly. In fact, it is not possible to solve this problem more efficiently. What some participants may have found difficult is coding this algorithm or approach so that it works correctly in all cases.

We need to carefully consider all possible starting locations of the word and each possible direction from each of these locations. The number of possibilities increases from subtask to later subtask.

Other than for the first subtask, the grid must be stored in memory. It is a two-dimensional structure so a data structure such as a list of lists (or array of arrays) is needed. It is easy to make errors "around the edge of the grid". Care must be taken to only access valid locations within the data structure of choice.

For this last subtask, the possibility of perpendicular bends in the hidden word presents a significant additional challenge because the number of possible locations for the hidden word is quite large. This can result in long repetitive code with lots of cases which is very error-prone. Carefully designed functions with well-chosen parameters can help remove lots of the redundancy reducing the likelihood of errors and simplifying the process of debugging if errors do occur.

The CENTRE for EDUCATION in MATHEMATICS and COMPUTING

cemc.uwaterloo.ca

Commentaires sur le CCI de niveau junior de 2023

L'objectif de ce commentaire est de donner un bref aperçu des éléments nécessaires pour résoudre chaque problème et des défis à relever. Les notes ci-dessous peuvent être utilisées pour guider toute personne qui souhaite tenter de résoudre ces problèmes. Or, elles n'ont pas pour but de décrire tous les détails d'une solution correcte. Nous encourageons toute personne intéressée à essayer d'élaborer une solution correcte en utilisant le langage de programmation de son choix.

J1 Livr-à-droïde

Pour évaluer les points gagnés en fonction du nombre de colis livrés et les points perdus en raison des collisions avec des obstacles, il est nécessaire de réaliser une opération de calcul sur les valeurs d'entrée. De plus, il faut utiliser une instruction conditionnelle pour déterminer si des points bonus seront accordés ou non.

J2 Piments

La résolution de ce problème peut être accomplie en utilisant une variable pour cumuler le piquant total du chili de Ron. Cette valeur peut être mise à jour pour chaque nom de piment lu à partir de l'entrée à l'aide d'une boucle qui se répète N fois. Pour effectuer cette mise à jour dans le corps de la boucle, on peut utiliser une instruction conditionnelle qui teste chaque nom de piment possible. Autrement, il est possible de stocker le tableau de l'énoncé du problème en mémoire et de consulter ce tableau pour déterminer la valeur SHU selon le nom du piment. Les différents langages de programmation permettent de stocker ce tableau de différentes manières.

J3 Événement spécial

Pour chaque sous-tâche de ce problème, on doit garder une trace du nombre de personnes pouvant participer à l'événement pour chaque jour; ce que l'on peut conserver dans cinq variables différentes étant donné qu'il n'y a que cinq jours. Toutefois, pour éviter les erreurs et rendre le code plus lisible, il est préférable de stocker ces données dans une liste ou un tableau.

Pour la première sous-tâche, nous pouvons procéder en recherchant le jour où le nombre de personnes est égal à N. Il est précisé qu'il n'y aura qu'un seul tel jour.

La deuxième sous-tâche consiste à déterminer le jour où le nombre de personnes présentes sera le plus élevé. Pour ce faire, il faut calculer parmi les cinq jours la valeur maximum du nombre de personnes pouvant participer à l'événement et identifier le jour correspondant à cette valeur maximale. Il ne peut y avoir qu'une seule journée avec ce nombre maximum.

Pour la troisième et dernière sous-tâche, on doit gérer les "égalités". C'est-à-dire qu'il peut y avoir plus d'un jour avec le même nombre maximum de personnes. Dans ce cas, il est délicat de produire la bonne réponse. Remarquons qu'il est important d'éviter d'imprimer une virgule après le dernier jour.

J4 Allée de triangles

Pour aider Bocchi à déterminer le nombre de mètres de ruban de signalisation dont elle aura besoin, il faut que l'on procède à un comptage minutieux.

Pour la première sous-tâche, on peut compter B, qui correspond au nombre de fois où 1 parait dans l'entrée. La réponse correcte sera $3 \times B$. Remarquons que l'on peut ignorer la deuxième ligne des données d'entrée, ce qui est également vrai pour la deuxième sous-tâche. Cependant, pour cette deuxième sous-tâche, il pourrait y avoir des triangles noirs adjacents. Dans ce cas, Bocchi n'a pas besoin de placer du ruban de signalisation sur le côté commun. Remarquons que chacun de ces côtés communs augmente de 2 la valeur de $3 \times B$. Cela signifie qu'on obtient la réponse finale correcte en soustrayant deux de $3 \times B$ pour chaque paire de triangles noirs adjacents.

Le même principe s'applique pour la troisième sous-tâche, mais on doit également rechercher les triangles noirs qui sont adjacents mais dans des rangées différentes. Ces triangles peuvent uniquement se trouver dans les positions paires (en supposant que l'on commence à compter à partir de zéro).

On peut résoudre ce problème sans nécessairement remarquer qu'on peut effectuer des soustractions à $3 \times B$. Cette approche consiste à rechercher tous les endroits où il faut placer du ruban de signalisation. Il faut utiliser un algorithme qui inclut les côtés des triangles qui sont situés sur les bords de l'allée ainsi que les côtés entre les triangles adjacents qui ne sont pas de la même couleur.

Quelle que soit l'approche adoptée, il existe plusieurs solutions aux trois premières sous-tâches qui sont très efficaces. Cependant, la quatrième sous-tâche a pour but de garantir cette efficacité en interdisant les solutions trop lentes. Elle impose en particulier que le nombre de passages sur chaque rangée ne soit pas trop élevé et précise que seul un nombre constant de passages est autorisé. Les solutions qui ne respectent pas cette contrainte sont probablement trop lentes car elles utilisent une fonction intégrée dans une boucle qui elle-même passe en boucle sur une rangée de triangles.

J5 Chasse aux mots CCI

Le défi présenté par le dernier problème du concours de cette année était d'une nature différente de celle des années précédentes. L'examen systématique de chaque emplacement possible du mot dans la grille permet d'obtenir tous les points si la tâche est correctement exécutée. En fait, il n'est pas possible de résoudre ce problème de manière plus efficace. Ce qui peut avoir posé problème à certains participants est le fait de coder cet algorithme ou cette approche de manière qu'elle fonctionne correctement dans tous les cas.

Il était nécessaire de considérer soigneusement tous les emplacements de départ possibles du mot, ainsi que chaque direction possible à partir de ces emplacements. Le nombre de possibilités augmente de sous-tâche en sous-tâche.

À part pour la première sous-tâche, la grille devrait être stockée en mémoire. Comme il s'agit d'une structure bidimensionnelle, une structure de données telle qu'un tableau d'arrays irréguliers (ou un tableau de tableaux) est nécessaire. Il est facile de faire des erreurs "au bord de la grille". Il faut faire attention à n'accéder qu'aux emplacements valides dans la structure de données choisie.

Pour cette dernière sous-tâche, la possibilité de courbes perpendiculaires dans le mot caché représente un défi supplémentaire important, car le nombre d'emplacements possibles pour le mot caché est grand. Cela peut résulter en un code long et répétitif comprenant de nombreux cas et étant donc très sujet aux erreurs. Des fonctions soigneusement conçues avec des paramètres bien choisis peuvent contribuer à éliminer une grande partie de la redondance, ce qui réduit la probabilité d'erreurs et simplifie le processus de débogage si des erreurs se produisent.