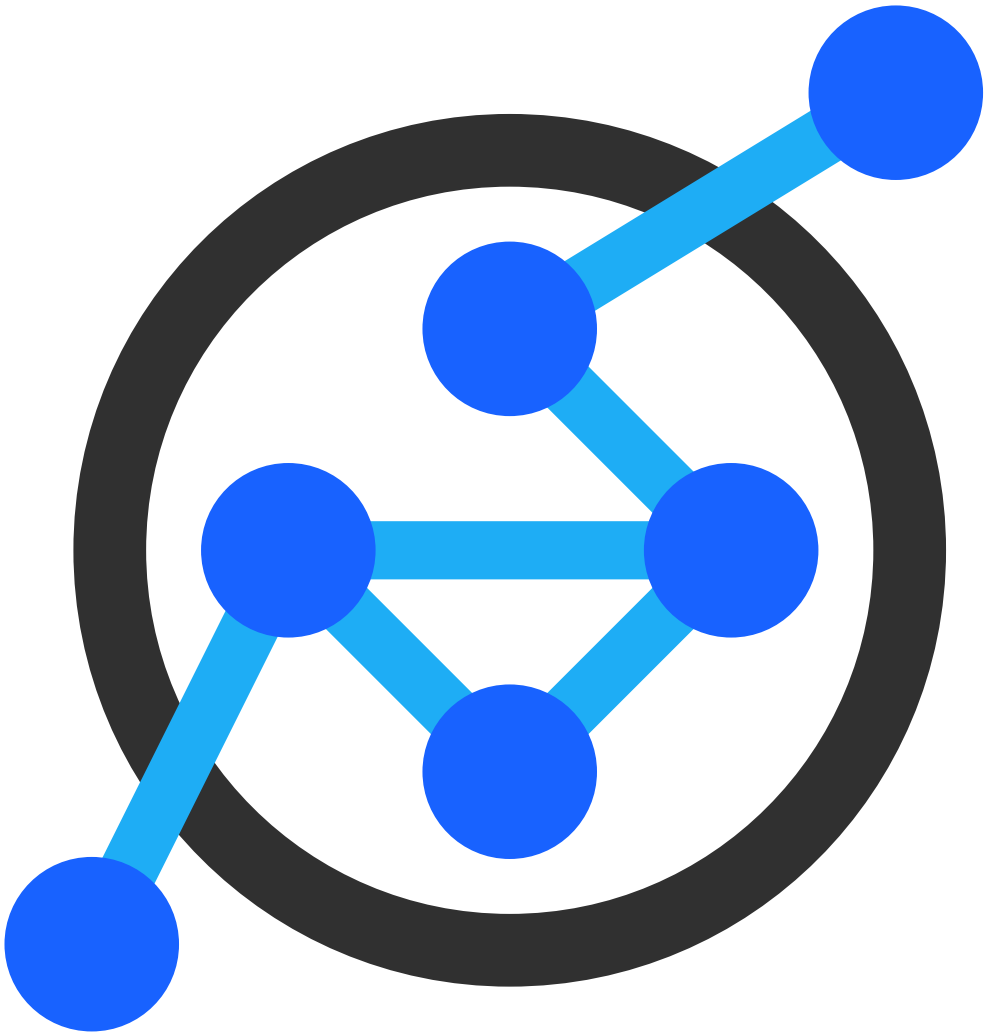
# Deuxième Tour Théorique

# Tâches



Swiss Olympiad in Informatics

4 – 7 mars 2022

# Instructions

— Tu as 5 heures pour résoudre les tâches. Réserve suffisamment de temps pour envoyer tes réponses. Nous recommandons de réserver 30 minutes pour le scan et la mise en ligne. Si tu finis en avance, tu peux continuer à travailler sur les tâches et envoyer tes solutions à nouveau plus tard.
— Le concours est à livres ouverts : tu peux utiliser internet, des livres, d'anciennes solutions, etc.
— Si tu as des questions pendant le concours, pose-les par e-mail (`info@soi.ch`). Nous ne donnerons pas d'indices, donc essaie de formuler des questions claires auxquelles il est possible de répondre par oui ou par non.
— Ta solution doit être entièrement rédigée à la main sur du papier.
— Écris lisiblement.
— La solution doit se suffire à elle-même : tu ne peux pas faire référence à des algorithmes connus, des blogs ou des articles, à l'exception des sections "Premier tour" et "Deuxième tour" du Wiki SOI.
— Envoie un PDF par tâche sur le grader, avec une assez bonne résolution pour que nous puissions tout lire. Quelques mégabytes par tâche devraient suffire (la limite est de 100 MiB).
— Nous ne consultons que la dernière soumission pour chaque tâche. Tu peux en envoyer autant que tu veux.
— Tu peux voir un aperçu de ce que tu as envoyé en cliquant sur "Detail", afin de vérifier que tu as envoyé le bon fichier.
— En cas de problème technique avec le scanner ou le grader, tu peux nous prouver que tu as fini à temps en remettant une photo à basse résolution. Envoie-la à Joël Huber.

## Tâches

Fais attention à l'ordre des tâches : la première est censée être la plus facile. Les trois autres tâches sont arrangées dans un ordre arbitraire. Pour cette raison, nous te recommandons de lire tous les énoncés et de réfléchir à propos de chaque tâche.

Dans chaque tâche, sauf mention contraire, tu peux supposer que l'entrée est donnée comme des variables et tableaux *en lecture seule*, qui *ne comptent pas* dans l'utilisation de mémoire de ta solution.

## Évaluation

Les solutions sont évaluées selon des critères similaires à ceux du premier tour théorique. Les deux critères les plus importants sont la correction et le temps d'exécution asymptotique. La qualité de la description et les arguments prouvant la correction sont également pris en compte.

Tu peux toujours te référer à du contenu du Wiki SOI ou du tour 2H. Tu ne dois pas expliquer, par exemple, pourquoi l'algorithme de Dijkstra fonctionne, mais tu dois expliquer pourquoi et comment il peut être appliqué. Dans le cas de Dijkstra, tu devrais expliquer clairement sur quel graphe tu l'utilises et noter que les poids des arêtes de celui-ci ne sont pas négatifs.

L'algorithme doit être décrit en suffisamment de détail pour qu'il soit aisé de convertir la description en un programme. Écris également quelles structures de données tu utiliserais. Habituellement, le mieux est d'écrire un bref pseudocode.

Si un algorithme t'est demandé, tu devrais utiliser cette structure comme ligne directrice :

1. Décris l'idée derrière un algorithme qui résout le problème.
2. Donne du pseudocode ou explique comment on pourrait implémenter l'algorithme.
3. Prouve par des arguments la correction de l'approche utilisée.
4. Indique et justifie le temps d'exécution asymptotique ainsi que l'utilisation de mémoire.

Si une partie de ta solution est utile pour plusieurs sous-tâches, il suffit de l'écrire une seule fois et de t'y référer à partir de là. Note, cependant, qu'un algorithme peut résoudre plusieurs sous-tâches mais pas nécessairement de façon optimale.

# Consensus

Pour Stofl la souris, le camp de Sbrinz est toujours le meilleur moment de l'année. Huit jours consacrés aux algorithmes, à la programmation et au fromage à pâte dure. Quoi de mieux ?

Hélas, pour Stofl, le camp est bien trop court. C'est pourquoi il est allé vers chacun des $n$ leaders pour leur demander quel est le nombre optimal de jours pour le camp selon eux. Comme on pouvait s'y attendre, Stofl a reçu des tas d'opinions différentes. Les leaders lui ont dit que s'il arrive à les faire parvenir à un consensus, c'est-à-dire à les rendre tous d'accord sur un même nombre de jours, ils changeraient la durée du camp pour la fixer à ce nombre. Stofl accepte leur défi.

Te sont donnés $n$ nombres $a_0, a_1, \ldots, a_{n-1}$, où $a_i$ est le nombre de jours préféré par le $i$-e leader.

Stofl peut *convoquer une médiation* entre exactement $k$ leaders (où $k$ est une constante donnée dans l'entrée) en les invitant à une discussion et en faisant l'une des choses suivantes :

- Faire un vote sur la question : le camp devrait-il durer $\geq x$ jours ? Au moins la moitié des leaders présents à la médiation (i.e. $\geq k/2$) devraient être d'accord pour que le vote soit accepté.
- Faire un vote sur la question : le camp devrait-il durer $\leq x$ jours ? Au moins la moitié des leaders présents à la médiation (i.e. $\geq k/2$) devraient être d'accord pour que le vote soit accepté.
- Si les deux votes sont acceptés, les leaders présents à la médiation changent leur opinion et pensent désormais que $x$ est le nombre de jours idéal pour le camp.

Stofl peut convoquer une médiation aussi souvent qu'il le souhaite, pourvu que cela finisse en une durée finie.

Quel est le nombre de jours maximal pour lequel Stofl peut atteindre un consensus ?

**Exemple** Entrée : $k = 3$, $a = [3, 1, 4, 1, 5]$.

- Dans le premier tour de médiation, Stofl pourrait inviter les leaders 0, 1 et 3 et les faire se mettre d'accord sur $x = 1$. Les opinions seraient désormais $a = [1, 1, 4, 1, 5]$.
- Dans le deuxième tour de médiation, Stofl inviterait les leaders 0, 2 et 4 et les ferait se mettre d'accord sur $x = 4$. Les opinions seraient désormais $a = [4, 1, 4, 1, 4]$.
- Dans le troisième tour de médiation, Stofl inviterait les leaders 0, 1 et 2 et les ferait se mettre d'accord sur $x = 4$. Les opinions seraient désormais $a = [4, 4, 4, 1, 4]$.
- Dans le quatrième tour de médiation, Stofl inviterait les leaders 0, 1 et 3 (comme au premier tour) et les ferait se mettre d'accord sur $x = 4$. Les opinions seront alors $a = [4, 4, 4, 4, 4]$.

Le consensus est alors 4. Pour cette entrée spécifique, c'est aussi le plus grand nombre possible.

**Exemple** Entrée : $k = 3$, $a = [3, 1, 4, 1, 5]$. Stofl pourrait obtenir un consensus pour soit 1, 2 ou 3.

## Sous-tâche 1: Résous l'exemple (10 points)

Pour l'entrée suivante, quel est le plus grand consensus que Stofl peut produire ? $a = [1, 6, 1, 8, 0, 3, 3]$ Résous ceci pour *a)* $k = 3$ et *b)* $k = 2$.

Écris, pour chaque tour de médiation, quels leaders tu invites, sur quel nombre ils se mettent d'accord et à quoi les opinions ressembleront à la sortie de la médiation. Tu n'as pas besoin de prouver que ton nombre est optimal.

## Sous-tâche 2: Un algorithme (90 points)

Aide Stofl à trouver un algorithme qui résout le problème pour tout $2 \leq k \leq n$ et $n \geq 2$. Ton algorithme doit seulement imprimer le nombre maximal de jours du camp (et pas les différents tours de médiation.

Prouve pourquoi ton algorithme est correct.

Indique le temps d'exécution et l'utilisation de mémoire en fonction de $n$ et $k$. Essaie d'optimiser le temps d'exécution en premier, la mémoire ensuite. Note que $k \leq n$, ce qui veut dire qu'un algorithme dont le temps d'exécution est $O(n^7 k)$ est considéré être plus rapide qu'un algorithme en $O(n^8)$, mais qu'un algorithme dont le temps d'exécution est $O(nk^2)$ est considéré être plus lent qu'un algorithme en $O(n^2)$ (par exemple, dans le cas où $n = k$).

Tu peux obtenir un score partiel en résolvant les sous-problèmes suivants :

*a)* $k = 3$ – les médiations ont toujours lieu entre 3 personnes (30 points).
*b)* $a_i = 8$ ou $a_i = 9$ pour tout $i$ – il y a au plus deux opinions différentes (20 points).

Si tu résous le problème complet, tu peux bien sûr ignorer ces sous-problèmes ; mais trouver un algorithme plus efficace pour les sous-problèmes peut te rapporter plus de points.

# Conférence scientifique

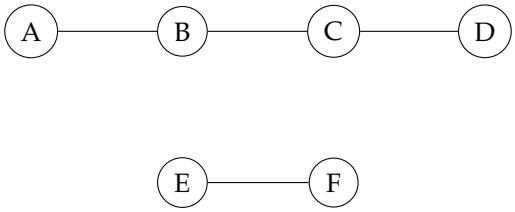
La souris Binna organise une conférence scientifique. Il y a $n$ souris scientifiques qu'elle pourrait inviter. Elle sait également que $m$ paires de scientifiques ont déjà publié ensemble.

Pour chaque scientifique qui est invité à la conférence, leur *popularité* à la conférence est définie par le nombre d'autres scientifiques qui sont invités à la conférence et qui ont déjà publié ensemble avec eux.

La souris Binna sait que s'il existe un scientifique invité tel que sa popularité plus un (on ajoute un pour prendre en compte le scientifique lui-même) est au moins la moitié du nombre total de scientifiques invités, alors les discussions durant la conférence vont être trop centrées sur leur travail. Cela serait mauvais pour la formation de nouvelles idées, et elle veut donc éviter que cela arrive.

Formellement, si la souris Binna invite $k$ souris scientifiques $a_1, a_2, \ldots, a_k$ à la conférence, alors pour tout $i$ il faut que $1 + \#\{j : 1 \leq j \leq k, j \neq i, \{a_i, a_j\} \in E\} < \frac{k}{2}$, où $E$ est l'ensemble des paires de scientifiques qui ont publié ensemble. Notez que ces paires ne sont pas ordonnées, donc $\{a_i, a_j\} \in E$ veut dire la même chose que $\{a_j, a_i\} \in E$.

Considère l'exemple suivant, où les cercles représentent des scientifiques et les scientifiques qui ont publié ensemble sont connectés par une ligne ou une courbe :




Inviter tous les scientifiques à la conférence ne fonctionnerais pas, car les scientifiques $B$ et $C$ aurait une popularité 2, et ainsi la contrainte décrite si dessus pour $B$ et $C$ serait $1 + 2 < \frac{6}{2}$, ce qui est faux. Cependant, on pourrait inviter les scientifiques $A$, $C$, $D$, $E$ et $F$. Alors, la popularité des scientifiques $C$, $D$, $E$ et $F$ serait 1, et la popularité du scientifique $A$ serait 0, et les deux contraintes $1 + 1 < \frac{5}{2}$ et $1 + 0 < \frac{5}{2}$ sont vraies.

La Souris Binna n'est pas intéressée par le nombre de scientifiques qui sont invités à la conférence, tant qu'il y en a au moins un : $k > 0$. Peux-tu l'aider à trouver quels scientifiques elle devrait inviter, ou bien déterminer qu'organiser une telle conférence est impossible ? Dans le cas où il y a plusieurs ensembles de scientifiques qui satisfont les contraintes ci-dessus, n'importe lequel de ces ensembles sera accepté.

De son expérience passée à organiser des conférences scientifiques, la souris Binna sait qu'il est plus facile de trouver un tel ensemble de scientifiques lorsque le nombre de paires de scientifiques qui ont publié ensemble est petit. Ainsi, dans ce problème, nous allons aussi donner des points pour les solutions qui fonctionnent uniquement dans les cas où le nombre $m$ est relativement petit et les solutions qui obtiennent une meilleure complexité dans leur temps d'exécution lorsque $m$ est relativement petit. Regarde la liste des sous-tâches ci-dessous pour les détails.

Pour chacune des sous-tâches ci-dessous, tu dois décrire un algorithme, écrire son *pseudocode*, la *complexité du temps d'exécution* et l'*utilisation de mémoire* de ton algorithme et expliquer pourquoi il est *correct*.

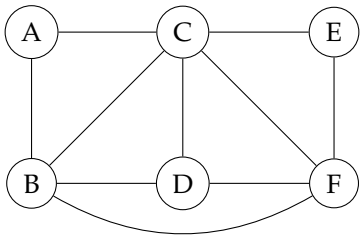
Pour avoir tous les points dans les sous-tâches 2-4, ta solution doit être correcte et avoir une bonne complexité du temps d'exécution et de l'utilisation de mémoire pour les cas où $m$ satisfait les limites dans la sous-tâche. Cependant, les réponses partielles et les solutions sous-optimales dans chaque sous-tâches seront également récompensées.

Tu es libre de proposer des solutions qui répondent à toutes ou plusieurs sous-tâches à la fois, ou de proposer des solutions différentes pour chacune des sous-tâches.

Dans toutes les sous-tâches, tu peux présumer que $n \geq 5$.

## Sous-tâche 1: Résous un exemple (10 points)

Pour l'ensemble de scientifiques suivant, où les cercles représentent des scientifiques et les scientifiques qui ont publié ensemble sont connectés par une ligne ou une courbe, peux-tu trouver un ensemble de scientifiques qui peuvent être invités à une conférence ?




Pour éviter tout malentendu, voici le même exemple en texte :

$$\{\{A,B\},\{A,C\},\{B,C\},\{B,D\},\{C,D\},\{C,E\},\{C,F\},\{D,F\},\{B,F\},\{E,F\}\}$$

Tu n'as pas besoin de prouver ta réponse, il suffit d'écrire un ensemble correct.

## Sous-tâche 2: Très peu de publications (20 points)

Le nombre de paires de souris scientifiques qui ont publié ensemble est très petit : $m < \frac{(n-5)^2}{10}$.

## Sous-tâche 3: Peu de publications (30 points)

Le nombre de paires de souris scientifiques qui ont publié ensemble est relativement petit : $\frac{(n-5)^2}{10} \leq m < \frac{(n-5)^2}{4}$.

## Sous-tâche 4: Beaucoup de publications (40 points)

Le nombre de paires de souris scientifiques qui ont publié ensemble est relativement grand : $m \geq \frac{(n-5)^2}{4}$.

# Portails à sens unique

Stofl la souris vient d'être élu ministre des Transports de l'Alliance des systèmes indépendants (l'*alliance* en abrégé).

L'Alliance est composée de $n$ planètes et la planète Apodemus est la capitale. Chaque planète était habitée par exactement 1 souris. Sous le nom de code "tous les portails mènent à Apodemus", le prédécesseur de Stofl avait lancé le projet de faire déménager toutes les souris vers la capitale en construisant $n-1$ portails à sens unique qui permettent d'atteindre Apodemus depuis toutes les planètes (soit directement, soit en passant par d'autres planètes). Au final, le ministre des Transports précédent n'a pas été réélu car il a échoué dans sa mission : bien que certaines planètes soient maintenant inhabitées, d'autres contiennent toujours une souris ayant refusé de déménager vers Apodemus.

Pour encourager les souris restantes à déménager, Stofl, en tant que nouveau ministre des Transports, veut construire jusqu'à $k$ portails à sens unique supplémentaires. Le but est de minimiser le nombre moyen de portails nécessaires pour déménager sur Apodemus, en prenant en compte chaque planète sur laquelle une souris habite encore. Stofl connaît la configuration des $n-1$ portails existants et sait quelles planètes sont habitées, mais il ne connaît pas le meilleur moyen d'ajouter ces $k$ portails supplémentaires. Peux-tu l'aider ?

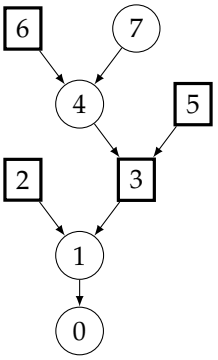
*Formellement*, l'Alliance peut être représentée comme une arborescence (un arbre enraciné) dirigée non pondérée avec $n$ sommets, dont toutes les arêtes sont dirigées vers la racine. La racine de cet arbre est la planète Apodemus. Un sous-ensemble $S$ de ces sommets est toujours habité par une souris sur chaque sommet (Apodemus n'est pas dans $S$). Tu peux ajouter jusqu'à $k$ arêtes dirigées non pondérées. Soit $f(v)$ la longueur du plus court chemin de $v$ jusqu'à la racine après avoir ajouté ces arêtes, et soit

$$A = \frac{\sum_{v \in S} f(v)}{|S|}$$

le nombre moyen de portails sur $S$ en utilisant ces arêtes. Trouve la plus petite valeur atteignable pour $A$.

## Sous-tâche 1: Résous l'exemple (10 points)

Stofl commence par te demander de résoudre un exemple pour vérifier que tu es capable de l'aider. Dans cet exemple, $n = 8$, $k = 2$, les planètes habitées sont $S = \{2, 3, 5, 6\}$ et Apodemus est le sommet 0.




Quelle est la plus petite valeur de $A$ possible et comment placerais-tu les portails pour l'atteindre ? Il n'est pas nécessaire de justifier ta solution.

## Sous-tâche 2: Portails chers, mauvais gouvernement (20 points)

Les portails sont très chers, et l'Alliance ne peut que se permettre d'en acheter un. C'est pourquoi $k = 1$ dans cette sous-tâche.

Décris un algorithme qui résout ce problème et montre qu'il est correct. Ton algorithme n'a besoin de calculer que $A$. Tu peux supposer que l'entrée est donnée dans un format pratique à utiliser (par exemple, Apodemus est le sommet 0 et l'arbre est donné sous forme d'une liste d'adjacence).

Analyse son temps d'exécution et son utilisation de mémoire en fonction de $n$.

## Sous-tâche 3: L'Alliance (70 points)

Dans cette sous-tâche, Stofl te demande finalement de l'aider à résoudre le problème complet.

Décris un algorithme qui résout ce problème et montre qu'il est correct. Comme ci-dessus, ton algorithme n'a besoin de calculer que $A$, et tu peux supposer que l'entrée est donnée dans un format pratique à utiliser.

Analyse son temps d'exécution et son utilisation de mémoire en fonction de $n$ et $k$. Essaie d'optimiser le temps d'exécution avant l'utilisation de mémoire.

Note que $k \leq n$, ce qui veut dire qu'un algorithme dont le temps d'exécution est $O(n^7 k)$ est considéré comme plus rapide qu'un algorithme en $O(n^8)$, mais qu'un algorithme dont le temps d'exécution est $O(nk^2)$ est considéré être plus lent qu'un algorithme en $O(n^2)$ (par exemple, dans le cas où $n = k$).

*Remarque* : si tu n'arrives pas à trouver un algorithme, des progrès significatifs vers une solution pourront remporter des points partiels.

# Détecteur de particules

Binna la souris veut faire une expérience de physique des particules au Grand Collisionneur de Hadrons, qui est situé profondément sous la surface terrestre près de Genève. À cette fin, elle a construit un détecteur de particules, composé d'un anneau de capteurs. Il y a $N$ capteurs numérotés comme $0, 1, \ldots, N-1$. Les capteurs 0 et 1 sont adjacents, comme les capteurs 1 et 2, …, $N-2$ et $N-1$, $N-1$ et 0.

Malheureusement, certains capteurs peuvent présenter un défaut de fabrication, qui n'est pas facile à détecter, et Binna ne sait pas quels capteurs sont cassés (il se peut même que tous le soient ou qu'aucun ne le soit). Les capteurs ont un mode spécial qui permet de détecter si un capteur adjacent est cassé. Si un capteur adjacent est cassé, il rapporte un 1, sinon un 0. Cependant, si les deux capteurs adjacents sont en panne, ils s'annulent mutuellement et le défaut ne peut être détecté. De plus, si le capteur avec lequel la panne est détectée est lui-même en panne, le résultat est inversé.

Essentiellement, cela signifie qu'un bon capteur indiquera 0 si les deux voisins ont le même degré de fonctionnement (soit les deux sont bons, soit les deux sont cassés), et 1 sinon. De même, un capteur cassé rapportera 1 si les deux voisins ont le même degré de fonctionnement, et 0 sinon.

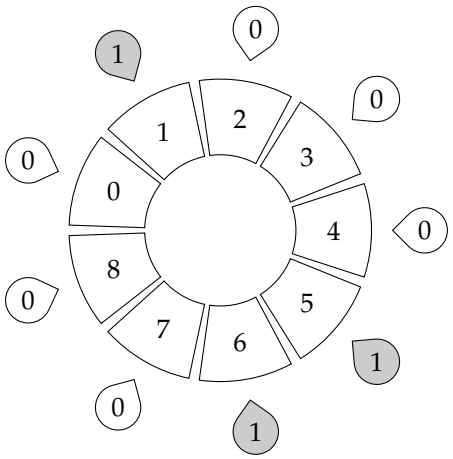
Binna a collecté les rapports de tous les capteurs. Elle veut maintenant savoir combien de capteurs défectueux il peut y avoir au maximum. Il se peut également qu'il n'existe pas de configuration de capteurs en panne et en bon état permettant d'obtenir les valeurs indiquées, ce qui indique que le détecteur de particules présente un autre problème.

**Description formelle** Te sont donnés le nombre de capteurs $N \geq 3$ ainsi que l'indication $r_i \in \{0, 1\}$ sur chaque capteur $i \in \{0, \ldots, N-1\}$. Imprime le nombre maximal possible de capteurs cassés, ou $-1$ s'il n'y a pas de configuration de capteurs qui pourrait générer de telles indications.

## Sous-tâche 1: Un jouet-détecteur (15 points)

Binna a construit un petit détecteur qui ressemble à ceci. Les nombres dans les bulles sont les indications sur les capteurs.

Peux-tu déterminer combien de capteurs cassés il peut y avoir au plus ?



## Sous-tâche 2: Tous les capteurs indiquent 1 (10 points)

Tous les capteurs du détecteur indiquent 1.

Calcule le plus grand nombre possible de capteurs cassés pour n'importe quel $N$, pour le cas où $r_i = 1$ pour tout $i$. Écrire une expression suffit, il n'y a pas besoin de décrire un algorithme. Montre pourquoi ta réponse est correcte.

## Sous-tâche 3: Tous les capteurs indiquent 0 (15 points)

Tous les capteurs indiquent 0. Cela signifie-t-il que tout va bien ?

Calcule le plus grand nombre possible de capteurs casés pour n'importe quel $N$, pour le cas où $r_i = 0$ pour tout $i$. Écrire une expression suffit, il n'y a pas besoin de décrire un algorithme. Montre pourquoi ta réponse est correcte.

## Sous-tâche 4: Cas général (60 points)

Il n'y a plus aucune restriction. Décris ton algorithme.