Задачи Посылки

A. Тетрис с данными (10)

Ограничение времени	1 секунда
Ограничение памяти	64 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Задача предоставлена Научно-образовательным центром когнитивного моделирования МФТИ

В контексте машинного обучения, особенно при работе с последовательностями, необходимо эффективно обрабатывать данные переменной длины в параллельных вычислениях. Для этого можно дополнять короткие последовательности "пустыми" значениями до нужной длины, однако на это расходуются дополнительная память для хранения и ценные вычислительные ресурсы на обработку "пустых" значений. В качестве альтернативы можно использовать взаимодополнение последовательностей, чтобы свести "пустые" значения к минимуму или избежать их вовсе.

Ваша задача — упаковать n объектов разных размеров в минимальное количество "контейнеров" k фиксированной вместимости c.

Формат ввода

В первой строке дано значение c — вместимость каждого "контейнера" $\in [1, 50]$.

Во второй первой строке дано значение n — количество объектов $\in [1, 400]$

В третьей строке n значений через пробел — размер каждого объекта $\in [1, c]$.

Формат вывода

Минимальное количество "контейнеров" k, в которое можно упаковать данные объекты.

Пример

Ввод	Вывод
10 5 4 5 6 3 2	2

Задачи Посылки

B. Критический путь в микросхеме (10)

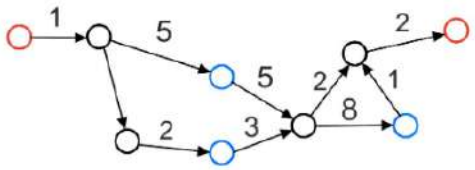
Ограничение времени	2 секунды
Ограничение памяти	64 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Задача предоставлена Научно-исследовательским институтом системных исследований Национального исследовательского центра "Курчатовский институт" (НИЦ "Курчатовский институт"-НИИСИ) при поддержке Российской нейросетевой ассоциации

Тактовая частота - ключевой параметр микросхемы, определяющий ее производительность. Чем выше тактовая частота схемы, тем больше «операций» может быть выполнено за единицу времени при прочих равных. Для вычисления тактовой частоты необходимо знать критический путь схемы.

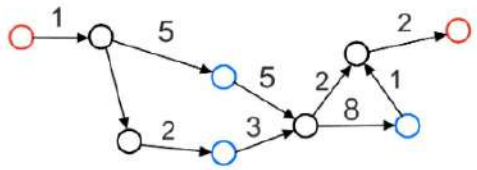
Критический путь - это самый длинный путь в схеме между двумя триггерам, либо триггером и портом, он начинается в одном триггере (или на входе схемы) и заканчивается в другом триггере (или на выходе схемы). Триггеры это небольшие ячейки памяти запоминающие данные на входе каждый такт, поэтому за такт сигнал должен успеть дойти от одного триггера до другого, иначе в работе схемы будут сбои. Другими словами, критический путь - это самое медленное звено в цепи вычислений, определяющее минимальное время между тактовыми импульсами при котором схема работает корректно.

Ваша задача реализовать алгоритм, который находит длину критического пути в схеме. Рассмотрим пример снизу, там красные вершины - порты, синие - триггеры, черные - остальные ячейки, формирующие критический путь (standard cells). Длина критического пути в этом примере равна 13, так как это самый длинный путь между триггерами, либо портом и триггером. Обратите внимание на то что критический путь должен содержать **только стандартные ячейки**, кроме начальной и конечной вершин, т.е. внутри критического пути не должно быть других триггеров.



Формат ввода

В первой строке задается число N - количество связей в схеме (ребер в графе). N не превышает 10^5.



Формат ввода

В первой строке задается число N - количество связей в схеме (ребер в графе). N не превышает 10^5.
Далее в каждой из N строк идет информация о направленных ребрах графа - 3 элемента через пробел: название source элемента, target элемента и задержка между ними. Названия элементов имеют следующий вид: ip_i - для входных портов (input ports), op_i - для выходных портов (output ports), ff_i - для триггеров (flip-flops) и sc_i - для стандартных ячеек (standard cells).
Считайте что все графы являются ациклическими, всегда хотя бы один путь существует.

Формат вывода

Одно число - длина критического пути.

Ваш ответ будет засчитан как правильный, если отличается от верного ответа не более чем на 10^{-3} по абсолютному значению.

Пример 1

Ввод	Вывод
6 ip_0 sc_0 0.25 sc_0 sc_1 0.9 sc_0 sc_2 0.7 sc_1 sc_3 0.14 sc_2 sc_3 1.68 sc_3 op_0 0.22	2.85

Пример 2

Ввод	Вывод
17 ip_0 sc_0 0 sc_0 ff_0 0.2 sc_0 sc_2 0.5	2.07

Одно число - длина критического пути.

Ваш ответ будет засчитан как правильный, если отличается от верного ответа не более чем на 10^{-3} по абсолютному значению.

Пример 1

Ввод	Вывод
6 ip_0 sc_0 0.25 sc_0 sc_1 0.9 sc_0 sc_2 0.7 sc_1 sc_3 0.14 sc_2 sc_3 1.68 sc_3 op_0 0.22	2.85

Пример 2

Ввод	Вывод
17 ip_0 sc_0 0 sc_0 ff_0 0.2 sc_0 sc_2 0.5 sc_0 sc_3 0.3 ff_0 sc_1 0.8 sc_2 sc_4 0.4 sc_3 sc_4 0.27 sc_1 sc_4 0.22 sc_4 sc_5 0.15 sc_4 ff_1 0.32 sc_5 ff_2 0.43 ff_1 sc_6 0.12 ff_2 sc_6 0.84 sc_6 op_0 0.12 sc_6 sc_7 0.9 sc_7 op_1 0.33 ip_1 sc_7 0	2.07

Пример 3

Ввод	Вывод
24 ip_0 sc_2 0 ip_0 sc_1 0	4.1

```
sc_3 sc_4 0.27
sc_1 sc_4 0.22
sc_4 sc_5 0.15
sc_4 ff_1 0.32
sc_5 ff_2 0.43
ff_1 sc_6 0.12
ff_2 sc_6 0.84
sc_6 op_0 0.12
sc_6 sc_7 0.9
sc_7 op_1 0.33
ip_1 sc_7 0
```

Пример 3

Ввод	Вывод
24 ip_0 sc_2 0 ip_0 sc_1 0 ip_0 sc_0 0.1 ip_1 sc_2 0.1 ip_2 sc_3 0 sc_2 sc_4 0.18 sc_2 sc_3 0.34 sc_1 sc_5 0.22 sc_0 sc_5 0.3 ip_4 sc_5 0.44 sc_5 ff_0 0.5 sc_5 op_0 0.65 sc_4 ff_0 0.45 sc_4 sc_6 0.1 sc_3 sc_4 0.2 ip_3 sc_6 0 sc_6 sc_7 0.15 sc_6 sc_9 0.15 ff_0 sc_7 1.8 sc_7 op_0 1.4 sc_7 sc_10 1.2 sc_9 sc_10 0.1 sc_9 op_1 0.2 sc_10 op_1 1.1	4.1

Язык: Python 3.12.3

Набрать здесь | Отправить файл

C. Грязные роботы (10)

Ограничение времени	1 секунда
Ограничение памяти	64 Мб
Ввод	стандартный ввод или input.txt
Вывод	стандартный вывод или output.txt

Задача предоставлена Институтом проблем управления им. В.А. Трапезникова РАН при поддержке Российской ассоциации искусственного интеллекта

На стройке работают N роботов. В конце каждого рабочего дня грязные роботы должны заехать на мойку, чистые – сразу отправиться в гараж. Каждый робот оснащен видеокамерой и лампочкой на макушке. С помощью видеокамеры робот может видеть и определить состояние (грязный или чистый) каждого другого робота, но не себя самого.

Все роботы собираются в общем холле. В холле установлена видеокамера на потолке, и рядом с ней – лампочка. Все роботы видят состояние лампочки. Если в холле есть хотя бы один грязный робот, лампочка загорается. После этого каждый робот должен определить свое состояние.

Время в мире роботов дискретно. На каждом шаге $t = 0, 1, \ldots, T-1$ все роботы синхронно производят одну попытку определить свое состояние на основе всех доступных на тот момент данных. Робот, который смог логически однозначно определить свое состояние, зажигает свою лампочку красным, если он грязный и белым, если он чистый.

Ваша задача — написать программу, имитирующую действия робота с номером N, исходя из доступной ему истории наблюдений. В конце каждого шага программа должна сказать, можно ли определить состояние робота, и вывести его, если можно. Программа одинаковая для всех роботов, и все роботы об этом знают.

Формат ввода

Стандартный ввод, первая строка содержит параметры задачи:

$N\ T\ r_1,\ldots,r_{N-1}$

$N \le 100$ – количество роботов;

$T \le 100$ – количество тактов наблюдений;

r_i - состояние робота $i = 1 \ldots N-1$, где $r_i = 0$, если робот i чистый и $r_i = 1$, если робот i грязный.

Следующие строки содержат наблюдаемые **перед** соответствующим тактом сигналы роботов для тактов с 0 по $T-1$:

$s_1, \ldots, s_{N-1}$

Если $s_i = -1$, то робот i на такте $t-1$ еще не знает свое состояние (лампочка не горит). Если $s_i = 1$, то робот i объявил, что он грязный. Если $s_i = 0$, то робот i объявил, что он чистый.

ли определить состояние робота, и вывести его, если можно. Программа одинаковая для всех роботов, и все роботы об этом знают.

Формат ввода

Стандартный ввод, первая строка содержит параметры задачи:

$N\ T\ r_{1},..,r_{N-1}$

$N \leq 100$ – количество роботов;

$T \leq 100$ – количество тактов наблюдений;

r_i - состояние робота $i = 1...N-1$, где $r_i = 0$, если робот i чистый и $r_i = 1$, если робот i грязный.

Следующие строки содержат наблюдаемые **перед** соответствующим тактом сигналы роботов для тактов с 0 по $T-1$:

$s_1, \ldots, s_{N-1}$

Если $s_i = -1$, то робот i на такте $t-1$ еще не знает свое состояние (лампочка не горит). Если $s_i = 1$, то робот i объявил, что он грязный. Если $s_i = 0$, то робот i объявил, что он чистый.

Формат вывода

Для каждой строки входных данных программа должна вывести одно число:

-1, если робот еще не может определить свое состояние на шаге t;

0, если робот узнал, что он чистый;

1, если робот узнал, что он грязный.

Система оценивания

Каждый тестовый файл содержит примеры входных данных для робота. Для каждой строки наблюдений из тестового файла программа должна вывести правильный сигнал робота.

Пример 1

Ввод	Вывод
2 2 1 -1 1	-1 0

Пример 2

Ввод	Вывод
3 2 1 0	-1

Система оценивания

Каждый тестовый файл содержит примеры входных данных для робота. Для каждой строки наблюдений из тестового файла программа должна вывести правильный сигнал робота.

Пример 1

Ввод	Вывод
2 2 1 -1 1	-1 0

Пример 2

Ввод	Вывод
3 2 1 0 -1 -1 -1 -1	-1 1

Пример 3

Ввод	Вывод
4 3 1 1 0 -1 -1 -1 -1 -1 -1 1 1 -1	-1 -1 0

Примечания

В примерах выше вывод программы смещен на одну строчку вверх.

Язык Python 3.12.3

Набрать здесь | Отправить файл

1

D. Построение GFlowNet модели (10)

Ограничение времени	2 секунды
Ограничение памяти	512 Мб
Ввод	стандартный ввод
Вывод	стандартный вывод

Введение

Эта задача посвящена моделям **Generative Flow Networks (GFlowNets)**, которые являются одним из методов генерации дискретных объектов.

Формально задача генерации дискретных объектов формулируется так: у нас есть конечное множество объектов $\mathcal{X}$ и неотрицательная reward функция $\mathcal{R}(x)$ определенная для всех $x \in \mathcal{X}$. Задача состоит в том, чтобы построить и обучить модель, которая генерирует объекты из $\mathcal{X}$ из вероятностного распределения $\mathcal{P}(x) = \frac{1}{Z}\mathcal{R}(x)$ для $Z = \sum_{x \in \mathcal{X}} \mathcal{R}(x)$.

Генеративный процесс с помощью GFlowNet может быть рассмотрен как последовательность действий, при которой мы стартуем из некоторого пустого объекта и на каждом шаге добавляем некоторый компонент в него.

Определим процесс формально:

- Рассмотрим ориентированный ациклический граф (DAG) $G = (\mathcal{S}, \mathcal{E})$, где $\mathcal{S}$ это множество состояний и $\mathcal{E} \subseteq \mathcal{S} \times \mathcal{S}$ это множество возможных переходов от одного состояния к следующему.
- В множество состояний $\mathcal{S}$ есть ровно одна вершина исток s_0 (стартовое состояние), в которую не входит ни одного ребра. Гарантируется, что все вершины графа G достижимы из s_0 по ребрам.
- В множестве состояний $\mathcal{S}$ содержится множество финальных объектов $\mathcal{X} \subseteq \mathcal{S}$, причем это множество в точности совпадает с множеством стоков графа (вершин, из которых не выходят ребра).
- Для каждого состояния $s \in \mathcal{S}$ рассмотрим множество возможных действий $\mathcal{A}_s$, а именно множество возможных состояний, в которые можно попасть из s. Формально $\mathcal{A}_s = \{s' \in \mathcal{S} | (s, s') \in \mathcal{E}\}$
- Генеративная модель GFlowNet $\mathcal{F}$ задается как множество распределений для действий из каждого состояния. Формально для всех состояний $s \in \mathcal{S}$ и следующих состояний $s' \in \mathcal{A}_s$ мы знаем неотрицательные числа $\mathcal{F}(s \to s')$, в соответствии с которыми будут генерироваться действия.

Определим процесс генерации финальных объектов из множества $\mathcal{X}$ с помощью заданной модели GFlowNet $\mathcal{F}$:

- Стартуем из стартового состояния $s := s_0$.
- До тех пор пока $s \notin \mathcal{X}$ генерируем следующее состояние $s' \in \mathcal{A}_s$ из вероятностного распределения $\mathcal{P}(s'|s) = \frac{1}{Z_s}\mathcal{F}(s \to s')$, где $Z_s = \sum_{s' \in \mathcal{A}_s} \mathcal{F}(s \to s')$. После этого делаем переход в сгенерированную вершину $s := s'$.
- Полученное финальное состояние $s \in \mathcal{X}$ в конце процесса генерации будет являться

Определим процесс генерации финальных объектов из множества $\mathcal{X}$ с помощью заданной модели GFlowNet $\mathcal{F}$:

- Стартуем из стартового состояния $s := s_0$.
- До тех пор пока $s \notin \mathcal{X}$ генерируем следующее состояние $s' \in \mathcal{A}_s$ из вероятностного распределения $\mathcal{P}(s'|s) = \frac{1}{Z_s}\mathcal{F}(s \to s')$, где $Z_s = \sum_{s' \in \mathcal{A}_s} \mathcal{F}(s \to s')$. После этого делаем переход в сгенерированную вершину $s := s'$.
- Полученное финальное состояние $s \in \mathcal{X}$ в конце процесса генерации будет являться сгенерированным объектом.

На практике GFlowNet-ы могут быть применены следующим образом. Обычно нам известно некоторое множество объектов $\mathcal{X}$ и reward функция $\mathcal{R}$ из реальной жизни (например, это могут быть геномные строки или молекулярные графы и reward функция взятая из их химических или физических свойств). Задача состоит в том, чтобы задать процесс конструирования объектов (граф $\mathcal{G}$) и построить для него модель $\mathcal{F}$ так, что описанный генеративный процесс будет генерировать объекты из желаемого распределения, заданного $\mathcal{R}$. Обычно в реальности множество объектов $\mathcal{X}$ очень большое и распределения $\mathcal{F}$ параметризуются нейронной сетью.

Задача

Вам полностью дан граф генеративных переходов $\mathcal{G}$, а также желаемое вероятностное распределение на финальных объектах $x \in \mathcal{X}$.

Постройте какую-нибудь GFlowNet модель $\mathcal{F}$, которая будет генерировать финальные объекты из заданного распределения.

Для полного балла также требуется, чтобы **по каждому ребру существовала ненулевая вероятность пройти**, то есть $\mathcal{F}(s \to s') > 0$ для всех $(s, s') \in \mathcal{E}$.

Формат ввода

В первой строке находится единственное целое число n ($2 \le n \le 50$) — количество состояний, то есть $n = |\mathcal{S}|$.

Для простоты пусть множество состояний будет $\mathcal{S} = \{1, 2, \ldots, n\}$, а стартовое состояние $s_0 = 1$.

В каждой из следующих n строк задаются возможные переходы из каждого состояния.

В строке с номером s сначала находится целое число k_s ($0 \le k_s \le n - s$) — количество возможных переходов из s, то есть $k_s = |\mathcal{A}_s|$. Затем следует k_s целых чисел $s'_1, s'_2, \ldots, s'_{k_s}$ ($s'_j > s$). Множество переходов задается как $\mathcal{A}_s = \{s'_1, s'_2, \ldots, s'_{k_s}\}$. Гарантируется, что заданные s'_i различны.

Заметим, что заданный ориентированный граф $\mathcal{G}$ является ациклическим (поскольку все ребра ведут из вершин с меньшим номером в вершины с большим номером). Гарантируется, что все вершины достижимы из вершины 1 по ребрам.

Множество финальных объектов $\mathcal{X}$ задается как множество стоков графа $\mathcal{G}$ (множество состояний s, для которых $|\mathcal{A}_s| = 0$).

В последней строке задано n целых чисел $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^6$). Гарантируется, что $w_i = 0$, если $i \notin \mathcal{X}$ и $w_i > 0$, иначе.

Желаемое вероятностное распределение на финальных объектах $x \in \mathcal{X}$ задается как $\mathcal{P}(x) = \frac{w_x}{\sum_{i=1}^{n} w_i}$.

Для полного балла также требуется, чтобы **по каждому ребру существовала ненулевая вероятность пройти**, то есть $\mathcal{F}(s \to s') > 0$ для всех $(s, s') \in \mathcal{E}$.

Формат ввода

В первой строке находится единственное целое число n ($2 \le n \le 50$) — количество состояний, то есть $n = |\mathcal{S}|$.

Для простоты пусть множество состояний будет $\mathcal{S} = \{1, 2, \ldots, n\}$, а стартовое состояние $s_0 = 1$.

В каждой из следующих n строк задаются возможные переходы из каждого состояния.

В строке с номером s сначала находится целое число k_s ($0 \le k_s \le n - s$) — количество возможных переходов из s, то есть $k_s = |\mathcal{A}_s|$. Затем следует k_s целых чисел $s'_1, s'_2, \ldots, s'_{k_s}$ ($s'_j > s$). Множество переходов задается как $\mathcal{A}_s = \{s'_1, s'_2, \ldots, s'_{k_s}\}$. Гарантируется, что заданные s'_i различны.

Заметим, что заданный ориентированный граф $\mathcal{G}$ является ациклическим (поскольку все ребра ведут из вершин с меньшим номером в вершины с большим номером). Гарантируется, что все вершины достижимы из вершины 1 по ребрам.

Множество финальных объектов $\mathcal{X}$ задается как множество стоков графа $\mathcal{G}$ (множество состояний s, для которых $|\mathcal{A}_s| = 0$).

В последней строке задано n целых чисел $w_1, w_2, \ldots, w_n$ ($0 \le w_i \le 10^6$). Гарантируется, что $w_i = 0$, если $i \notin \mathcal{X}$ и $w_i > 0$, иначе.

Желаемое вероятностное распределение на финальных объектах $x \in \mathcal{X}$ задается как $\mathcal{P}(x) = \frac{w_x}{\sum_{i=1}^{n} w_i}$.

Формат вывода

Выведите любую подходящую GFlowNet модель $\mathcal{F}$ в следующем формате:

Несколько строк, каждая строка содержит три **целых** числа s, s', $\mathcal{F}(s \to s')$ ($1 \le s, s' \le n$, $0 \le \mathcal{F}(s \to s') < 10^{300}$) — начало и конец ребра из графа $\mathcal{G}$ и значение GFlownet модели $\mathcal{F}$ для него. Для всех строк (s, s') должно быть ребром графа $\mathcal{G}$, для каждого ребра должно быть не более одной строки с этим ребром. Для всех не выведенных ребер (s, s') будет считаться, что $\mathcal{F}(s \to s') = 0$. Ребра можно выводить в любом порядке.

Если рассмотреть подграф, построенный на ребрах (s, s'), таких что $\mathcal{F}(s \to s') > 0$, то все вершины из $\mathcal{X}$ должны быть достижимы из 1, а также не должно существовать других стоков в нем. Для вершин, не достижимых в этом подграфе, можно не задавать значения $\mathcal{F}$ (они могут быть все нулевыми).

- Выведенная GFlowNet модель $\mathcal{F}$ должна генерировать в точности из заданного распределения. Если это не будет выполнено или ваш ответ не будет соответствовать формату в хотя бы одном из тестов, ваше решение получит 0 **баллов**.
- Если все ответы корректные, но в каком-то из тестов существует хотя бы одно ребро (s, s'), такое что $\mathcal{F}(s \to s') = 0$, то ваше решение получит 4 **балла**.
- Если во всех тестах для всех ребер $\mathcal{F}(s \to s') > 0$, ваше решение получит полные 10 **баллов**.

Можно показать, что в заданных ограничениях всегда существует ответ, получающий полный балл. Обратите внимание, что в данной задаче вам может потребоваться длинная

Желаемое вероятностное распределение на финальных объектах $x \in \mathcal{X}$ задается как $\mathcal{P}(x) = \frac{w_x}{\sum_{i=1}^{n} w_i}$

Формат вывода

Выведите любую подходящую GFlowNet модель $\mathcal{F}$ в следующем формате:

Несколько строк, каждая строка содержит три **целых** числа s, s', $\mathcal{F}(s \to s')$ ($1 \le s, s' \le n$, $0 \le \mathcal{F}(s \to s') < 10^{300}$) — начало и конец ребра из графа $\mathcal{G}$ и значение GFlownet модели $\mathcal{F}$ для него. Для всех строк (s, s') должно быть ребром графа $\mathcal{G}$, для каждого ребра должно быть не более одной строки с этим ребром. Для всех не выведенных ребер (s, s') будет считаться, что $\mathcal{F}(s \to s') = 0$. Ребра можно выводить в любом порядке.

Если рассмотреть подграф, построенный на ребрах (s, s'), таких что $\mathcal{F}(s \to s') > 0$, то все вершины из $\mathcal{X}$ должны быть достижимы из 1, а также не должно существовать других стоков в нем. Для вершин, не достижимых в этом подграфе, можно не задавать значения $\mathcal{F}$ (они могут быть все нулевыми).

- Выведенная GFlowNet модель $\mathcal{F}$ должна генерировать в точности из заданного распределения. Если это не будет выполнено или ваш ответ не будет соответствовать формату в хотя бы одном из тестов, ваше решение получит 0 **баллов**.
- Если все ответы корректные, но в каком-то из тестов существует хотя бы одно ребро (s, s'), такое что $\mathcal{F}(s \to s') = 0$, то ваше решение получит 4 **балла**.
- Если во всех тестах для всех ребер $\mathcal{F}(s \to s') > 0$, ваше решение получит полные 10 **баллов**.

Можно показать, что в заданных ограничениях всегда существует ответ, получающий полный балл. Обратите внимание, что в данной задаче вам может потребоваться длинная арифметика.

Пример

Ввод	Вывод
5 4 3 4 5 2 0 2 5 4 0 0 0 3 0 4 5	3 5 5 3 4 4 1 3 9 1 4 4 1 5 5 1 2 6

Скачать условие задачи

Язык: Python 3.12.3

Набрать здесь | Отправить файл