Задача Подкуп 2

⏳0 sec. 💾 0 MB

Задачата е тип output-only. В края на условието е описана по-подробна информация.

Боби има свързан двупосочен граф без мултиребра, но този път знае какъв е и за съжаление няма да се налага да подкупва комисията, за да го разбере! Приликата, обаче, е че Боби отново ще търси специално покриващо дърво на графа си.

Целта на Боби е да намери покриващо дърво на графа, което е възможно най-близо до това да има сумарно тегло на ребрата по-голямо от това на 67% от всички покриващи дървета. Тъй като може да има много покриващи дървета с еднакви тегла, за дадено покриващо дърво считаме половината от покриващите дървета с неговото сумарно тегло като такива с по-малко сумарно тегло. Оценяването е описано по-подробно в раздел *Оценяване*.

Напишете програма **`bribe2`**, която намира това покриващо дърво.

Вход

Всеки тест се състои от t подтеста. На първия ред се въвежда t. На първия ред на всеки подтест се въвеждат N и M – съответно броят върхове и ребра в графа. На следващите M реда се въвеждат по три числа u_i, v_i, w_i, което показва, че има двупосочно ребро между u_i и v_i с тегло w_i.

Изход

На t реда от стандартния изход се извеждат по $N - 1$ числа – индексите на ребрата, образуващи намереното покриващото дърво за всеки подтест. Индексите започват от 1.

Ограничения

- $1 \leq t \leq 10$
- $1 \leq N \leq 5\ 000$
- $1 \leq M \leq 10\ 000$
- $1 \leq u_i, v_i \leq N$, $u_i \neq v_i$
- $1 \leq w_i \leq 10\ 000$

Оценяване

Резултатът за всеки тест е равен на минималния от резултатите на подтестовете. Резултатът на подтест се смята по следния начин:

Нека $\mathcal{T}$ е множеството на всички различни покриващи дървета на графа, $k = |\mathcal{T}|$, а сумарното тегло на покриващо дърво T е $w(T) = \sum_{e \in T} w(e)$.

Нека T е покриващото дърво, изведено от Вас и нека

$$p = \frac{|\{T' \in \mathcal{T} \ : \ w(T') < w(T)\}| + \frac{1}{2}\,|\{T' \in \mathcal{T} \ : \ w(T') = w(T)\}|}{k},$$

т.е. p е делът на покриващите дървета в графа, чието сумарно тегло е по-малко от Вашето, като покриващите дървета със същото сумарно тегло като Вашето се броят с тежест $\frac{1}{2}$.

Ако T не е валидно покриващо дърво, резултатът за подтеста е 0. В противен случай резултатът е

$$s = \max\left(0,\ 1 - \frac{|p - 0.67|}{0.33}\right).$$

Качване на изходните файлове

Името на изходния файл за тест с номер xx трябва да бъде

test.xx.out

За тестове от 1 до 9 номерът се записва с водеща нула. Например изходът за осмия тест трябва да бъде във файл test.08.out, а изходът за десетия – във файл test.10.out.

На системата се качва ZIP архив, който съдържа поне един изходен файл. Изходните файлове трябва да се намират директно в архива, а не в поддиректория. Файлове с други имена се игнорират. Ако в архива липсва файл за даден тест, това качване не променя резултата Ви за този тест.

Всяко качване на архив се счита за един събмит, независимо колко изходни файла съдържа архивът. Ако качите изходи за един и същи тест в повече от един събмит, за крайното класиране се взема най-добрият резултат, постигнат от Вас на този тест. Невалиден изход носи 0 точки за съответния тест, но не оказва влияние, ако имате по-добри резултати за същия тест от предишен събмит.

За улеснение, можете да използвате следния код, който отваря всеки от тестовите файлове в съответния поток (стандартен вход/изход), след което извиква функция `solve`:

```
for(int test = 1; test <= 10; test++) {
    string s = (test<10?"0":"") + to_string(test);
    freopen(("test."+s+".in").c_str(), "r", stdin);
    freopen(("test."+s+".out").c_str(), "w", stdout);
    solve();
}
```

Така функцията `solve` може да чете входните данни чрез `cin` и да извежда изходните данни чрез `cout`.