Paweł Parys
Treść zadania, Opracowanie

Piotr Niedźwiedź
Program

Dostępna pamięć: 64 MB.

OI, Etap I, 20.10–17.11.2008

Gaśnice

Bajtazar zbudował nowy pałac. Pałac ten składa się z n pokoi i $n-1$ łączących je korytarzy. Pokoje są ponumerowane od 1 do n. Do pałacu jest jedno wejście, prowadzące do pokoju nr 1. Każdy korytarz łączy dwa różne pokoje. Do każdego pokoju prowadzi od wejścia dokładnie jedna droga (bez zawracania). Inaczej mówiąc, pokoje i łączące je korytarze tworzą **drzewo** *— spójny graf acykliczny.*

Inspektor straży pożarnej, który odbierał pałac, domaga się umieszczenia w pałacu gaśnic. Określił on następujące wymagania:

- *Gaśnice mają być umieszczone w niektórych pokojach, przy czym w jednym pokoju może znaleźć się kilka gaśnic.*
- *Każdemu pokojowi trzeba przydzielić jedną konkretną gaśnicę.*
- *Każda z gaśnic może być przydzielona do gaszenia co najwyżej s różnych pokoi.*
- *Dotarcie z dowolnego pokoju do przypisanej mu gaśnicy może wymagać przejścia co najwyżej k korytarzy.*

Bajtazar, jak to zwykle bywa po zakończeniu budowy, ma bardzo mało pieniędzy. Zastanawia się więc, jaka minimalna liczba gaśnic wystarczy do spełnienia powyższych wymagań?

Wejście

W pierwszym wierszu standardowego wejścia znajdują się trzy liczby całkowite n, s i k pooddzielane pojedynczymi odstępami, $1 \leqslant n \leqslant 100\,000$, $1 \leqslant s \leqslant n$, $1 \leqslant k \leqslant 20$. Każdy z następnych $n-1$ wierszy zawiera po dwie liczby całkowite oddzielone pojedynczym odstępem. W wierszu $i+1$ znajdują się liczby $1 \leqslant x_i < y_i \leqslant n$ reprezentujące korytarz łączący pokoje nr x_i i y_i.

Wyjście

W pierwszym i jedynym wierszu standardowego wyjścia powinna zostać wypisana jedna liczba całkowita — minimalna liczba gaśnic, jakie należy zainstalować w pałacu.

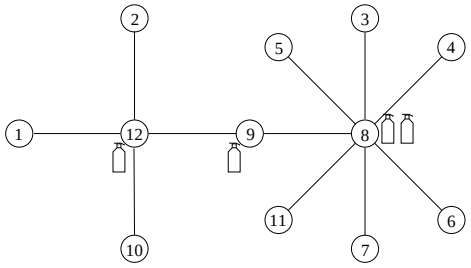
Przykład

Dla danych wejściowych:

```
12 3 1
1 12
3 8
7 8
8 9
2 12
10 12
9 12
4 8
5 8
8 11
6 8
```

poprawnym wynikiem jest:

```
4
```



Rozwiązanie

W tym zadaniu, podobnie jak w wielu innych (choć bynajmniej nie we wszystkich!), dobre efekty przynosi postępowanie zachłanne, czyli optymalizowanie ustawienia gaśnic lokalnie, w każdym fragmencie drzewa z osobna. Co to konkretnie oznacza? Drzewo będziemy przetwarzać od liści do korzenia. Intuicyjnie, w każdym poddrzewie[1] chcemy postawić jak najmniej gaśnic, a te, które stawiamy, chcemy postawić jak najwyżej, żeby obejmowały jeszcze jak najwięcej wierzchołków w górę.

Drzewo opisujące pałac Bajtazara nie ma naturalnego korzenia. Jako korzeń możemy wybrać pokój numer 1 (do którego prowadzi wejście do pałacu), ale równie dobrze może to być dowolny inny wierzchołek. Ważne, aby jakiś korzeń wybrać i ustalić go raz na zawsze. Jeśli algorytm oblicza minimalną liczbę potrzebnych gaśnic, to z definicji jego wynik nie zależy od wyboru korzenia. Jednak samo rozmieszczenie gaśnic już będzie zależało od tego, który wierzchołek jest korzeniem.

Przejdźmy teraz do konkretów. Zanim opiszemy algorytm, zastanówmy się, jakie informacje na temat rozmieszczenia gaśnic w poddrzewie są potrzebne „na zewnątrz", w pozostałej części drzewa. Na pewno ważne jest, ile gaśnic zostało rozstawionych. Nie ma znaczenia, gdzie dokładnie one stoją i do których pokoi są przypisane. Potrzebujemy jednak wiedzieć, czy blisko korzenia poddrzewa znajdują się jakieś nie do końca wykorzystane gaśnice, na jakiej są one głębokości i do ilu jeszcze pokoi możemy je przypisać. Dobrze jest wyobrażać sobie, że gaśnica ma s „końcówek", z których prowadzimy do pokoi ścieżki długości co najwyżej k. Niezbędna jest też informacja, ile jeszcze pokoi w poddrzewie nie ma przypisanej gaśnicy i na jakiej są głębokości.

Rozmieszczenie gaśnic w danym poddrzewie będziemy zatem charakteryzować przez następujący układ liczb: $(g, x_0, x_1, \ldots, x_k, y_0, y_1, \ldots, y_k)$, przy czym:

[1] Pod pojęciem *poddrzewa* będziemy w tym opisie rozumieli dowolny wierzchołek drzewa wraz ze wszystkimi jego potomkami.

- g to liczba gaśnic rozmieszczonych w tym poddrzewie
- x_i to liczba wolnych końcówek gaśnic, które sięgają na i pokoi w górę ponad korzeń
- y_i to liczba pokoi na głębokości i (korzeń znajduje się na głębokości 0), które potrzebują przypisania gaśnicy.

Dla przykładu, jeśli ustawimy gaśnicę o 1 pod korzeniem (tzn. w dziecku korzenia) i nie przypiszemy do niej żadnego pokoju, to x_{k-1} będzie równe s, gdyż gaśnica ta sięga jeszcze o $k-1$ pokoi ponad korzeń; jeśli już przypisane jest do niej jakieś j pokoi, to x_{k-1} będzie równe $s-j$. Zauważmy w szczególności, że (jeśli rozstawienie jest poprawne) $y_k = 0$, ponieważ pokojom na głębokości k nie możemy przypisać gaśnicy spoza poddrzewa (zatem y_k jest w naszym ciągu tylko dla wygody).

Gdy przeprowadzamy obliczenia w jakimś wierzchołku, to rozstawienia dla wszystkich poddrzew odpowiadających dzieciom tego wierzchołka powinny być już wyznaczone (drzewo przetwarzamy od liści do korzenia). Algorytm musi działać tak, żeby w tym wierzchołku umieścić jak najmniej gaśnic, żeby pokoje, do których gaśnica nie jest przypisana, były położone jak najbliżej korzenia oraz żeby wolne końcówki gaśnic sięgały jak najwyżej. Po chwili zastanowienia dochodzimy do wniosku, że algorytm musi (z dokładnością do drobnych szczegółów) działać w następujący sposób:

1. Najpierw sumujemy charakteryzacje wszystkich poddrzew; robimy to po współrzędnych, tzn. nowe g to suma wszystkich g z poddrzew, nowe x_0 to suma wszystkich x_0 z poddrzew itd.

2. Przesuwamy ciągi (ponieważ korzeń poddrzewa jest teraz o 1 wyżej niż poprzednio):

   - za nowe $x_0, x_1, \ldots, x_{k-1}$ przyjmujemy $x_1, x_2, \ldots, x_k$
   - za $y_1, y_2, \ldots, y_k$ przyjmujemy $y_0, y_1, \ldots, y_{k-1}$
   - za x_k przyjmujemy 0 (w korzeniu nie stoi na razie żadna gaśnica, więc nie ma wolnych końcówek wystających o k w górę)
   - za y_0 przyjmujemy 1 (jest jeden pokój na głębokości 0, który potrzebuje przypisania końcówki gaśnicy, a mianowicie korzeń).

   Zapominamy o starym x_0 (te wolne końcówki i tak nie sięgają w górę, nawet do nowego korzenia) oraz o starym y_k (które i tak jest równe zeru).

3. W korzeniu aktualnie badanego poddrzewa stawiamy $c = \lceil y_k/s \rceil$ gaśnic (czyli tyle, ile jest bezwzględnie koniecznych), tzn. wykonujemy instrukcje:

$$g \mathrel{+}= c \quad \text{oraz} \quad x_k = s \cdot c.$$

4. Dla każdego $i = 0, 1, \ldots, k$ parujemy nieprzypisane pokoje na głębokości i z wolnymi końcówkami wystającymi o i w górę. Możemy to zrobić, gdyż jeśli końcówka wystaje o i ponad korzeń, to sięga też na głębokość i. Innymi słowy, dla $c = \min(x_i, y_i)$ wykonujemy:

$$x_i \mathrel{-}= c \quad \text{oraz} \quad y_i \mathrel{-}= c.$$

5. Dla każdego $i < k$ parujemy nieprzypisane pokoje na głębokości i z wolnymi końcówkami wystającymi o $i+1$ w górę. Zatem dla $c = \min(x_{i+1}, y_i)$ wykonujemy instrukcje:
$$x_{i+1} \mathrel{-}= c \quad \text{oraz} \quad y_i \mathrel{-}= c.$$
Ważne, że wykonujemy ten krok dopiero po kroku 4.

W ten sposób otrzymujemy wynikowy ciąg dla naszego poddrzewa, który przekazujemy w górę. Zauważmy, że będzie w nim zachodziło $y_k = 0$, tak jak chcieliśmy, gdyż po wykonaniu kroku 3 zachodzi $x_k \geqslant y_k$.

W korzeniu całego drzewa zamiast kroków 3-5 trzeba oczywiście postąpić nieco inaczej:

3'. Teraz przeglądamy kolejne $i \in \{0, \ldots, k\}$ (będą to głębokości pokoi, które nie mają przypisanej gaśnicy) i dla każdego i przeglądamy po kolei wszystkie $j \in \{i, \ldots, k\}$ (długości wystających końcówek gaśnic). Ważna jest tutaj kolejność — zaczynamy od najmniejszych j, czyli od końcówek wystających jak najmniej w górę; te wystające bardziej mogą być także przydatne dla kolejnych i. Dla każdej takiej pary łączymy jak najwięcej pokoi z gaśnicami, czyli wykonujemy instrukcje
$$y_i \mathrel{-}= c \quad \text{oraz} \quad x_j \mathrel{-}= c$$
dla $c = \min(y_i, x_j)$.

4'. Pozostałych końcówek nie da się wykorzystać, więc w korzeniu drzewa stawiamy nowe gaśnice. Potrzeba ich $c = \lceil (y_0 + \ldots + y_k)/s \rceil$; gaśnice te przypisujemy wszystkim nieobsłużonym pokojom, tzn.
$$g \mathrel{+}= c, \quad x_k \mathrel{+}= s \cdot c - (y_0 + \ldots + y_k), \quad y_0 = \ldots = y_k = 0.$$

Implementując wprost powyższe rozwiązanie, dostajemy złożoność $O(nk)$. Autor zadania oraz jurorzy Olimpiady nie znają żadnego rozwiązania mającego lepszą złożoność (aczkolwiek nie jest wykluczone, że takie istnieje).

Intuicyjnie jasne jest, że opisane powyżej postępowanie daje najmniejszą możliwą liczbę gaśnic. Spróbujmy to jednak udowodnić formalnie. Dowód jest indukcyjny — dla coraz to większych poddrzew p będziemy dowodzić równocześnie, że:

A) Każde poprawne rozmieszczenie gaśnic w drzewie zawiera w poddrzewie p przynajmniej tyle gaśnic, ile zwraca nasz algorytm.

B) Mając poprawne rozmieszczenie gaśnic w całym drzewie, możemy przeorganizować gaśnice w poddrzewie p tak, aby stały zgodnie z naszym algorytmem. W reszcie drzewa ustawienie powinno pozostać bez zmian. Jedynie jeśli w p jest więcej gaśnic niż obliczył to nasz algorytm, to zbyteczne gaśnice wynosimy o jeden wierzchołek ponad korzeń p. Zgodność ma dotyczyć także przypisań pokoi do końcówek.

   Żeby być precyzyjnym, należy wyobrażać sobie przypisania pokoi końcówkom jako ścieżki długości co najwyżej k, jednak niekoniecznie ścieżki proste. Ścieżka taka biegnie najpierw w górę, a potem w dół. Nasz algorytm pewne przypisania wykonuje w obrębie poddrzewa p. Ścieżki od pozostałych, nieprzypisanych pokoi i nieprzypisanych końcówek prowadzi w górę, ponad p. Gdzieś tam poza p mogą

one zostać ze sobą połączone, ale takie przypisanie traktujemy już jako przypisanie poza p. Tutaj chcemy, żeby przypisania wykonane przez nasz algorytm wewnątrz p były w rozważanym rozmieszczeniu wykonane w ten sam sposób. Natomiast jeśli jakieś pokoje lub końcówki wychodzą w naszym algorytmie poza p, to w otrzymanym rozmieszczeniu też mają wychodzić poza p.

Baza indukcji (liść drzewa) jest trywialna. Natomiast aby dowieść powyższe dla jakiegoś poddrzewa p, najpierw stosujemy założenie indukcyjne do poddrzew tego poddrzewa odpowiadających dzieciom korzenia i przestawiamy w nich gaśnice tak, aby stały zgodnie z naszym algorytmem.

Zauważmy, że w korzeniu p trzeba postawić co najmniej $\lceil y_k/s \rceil$ gaśnic (tyle ile stawia nasz algorytm), bo pokojom na głębokości k trzeba przypisać gaśnicę, która stoi dokładnie tutaj. To już dowodzi własności A.

Proponuję Ci, Drogi Czytelniku, w tym miejscu przerwać czytanie i samodzielnie udowodnić własność B. Dowód nie jest trudny, wymaga jednak analizy różnych możliwych przypadków i przekonania się, że w każdym z nich wszystko jest w porządku. Jeśli jednak wolałbyś dowód ten przeczytać, zamieszczamy go poniżej.

Zauważmy wpierw, że rozstawienie gaśnic w poddrzewie p jest prawie wszędzie takie, jak powinno być; jedynie w korzeniu p może być więcej gaśnic niż ustawiłby tam nasz algorytm. Podobnie z przypisaniami pokoi do gaśnic — interesują nas jedynie te, które przechodzą przez korzeń p, gdyż pozostałe są już poprawione z założenia indukcyjnego.

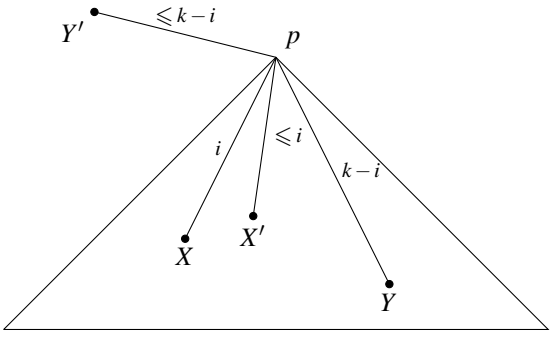
Poczyńmy na początku następujące spostrzeżenie: jeśli w rozważanym rozmieszczeniu istnieje połączenie pewnego pokoju na głębokości i z pewną końcówką wystającą na wysokość j, a nasz algorytm też wymaga połączenia pewnego pokoju na głębokości i z pewną końcówką o długości j, ale być może innych, to możemy połączenia tych pokoi i końcówek pozamieniać tak, aby prowadziły zgodnie z naszymi oczekiwaniami. To samo dotyczy pokoi lub końcówek na ustalonym poziomie niepołączonych z niczym wewnątrz p. Innymi słowy, nie liczy się, co dokładnie z czym jest połączone, liczy się tylko liczba połączeń danego rodzaju (rodzaj jest charakteryzowany przez głębokość łączonego pokoju i długość łączonej końcówki lub też brak jednej z tych dwóch rzeczy).

To pozwala już poradzić sobie z nadmiarowymi gaśnicami w korzeniu p. Przypuśćmy, że w korzeniu p jest c gaśnic. Początkowo pokoje na głębokości k mogą być podpięte do dowolnych z nich. Możemy jednak pozamieniać połączenia tak, żeby tylko $\lceil y_k/s \rceil$ gaśnic było połączonych z pokojami na głębokości k, pozostałe tylko z pokojami na mniejszej głębokości oraz z pokojami spoza p. Wówczas te $c - \lceil y_k/s \rceil$ nadmiarowych gaśnic możemy przesunąć o 1 w górę, uzyskując w korzeniu p tyle gaśnic, ile ustawiłby tam nasz algorytm.

Pozostaje naprawić przypisania. Bierzemy kolejno pokoje i staramy się je przypisać zgodnie z tym, jak zostałoby to wykonane w naszym algorytmie. Jeżeli natrafiamy na jakąś niezgodność, to musi wystąpić jeden z kilku przypadków:

- Mamy pokój, który zgodnie z algorytmem nie ma być przypisany niczemu w p, a teraz jest czemuś przypisany. W takim przypadku długość ścieżki od gaśnicy do tego pokoju jest mniejsza niż $k-1$, inaczej nasz algorytm wykonałby takie połączenie (jak już wspominaliśmy: niekoniecznie akurat tego wierzchołka z tą końcówką, ale między wierzchołkiem i gaśnicą na takich głębokościach). Zatem możemy nasze przypisanie wydłużyć o 2 tak, żeby wystawało poza p. To jest to, czego trzeba — nie mamy już przypisania wewnątrz p.

- Mamy pokój X na głębokości i, który zgodnie z algorytmem ma być przypisany końcówce Y wystającej o i w górę (czyli gaśnicy na głębokości $k-i$), a teraz jest przypisany pewnej innej końcówce Y' (przy czym X i Y znajdują się wewnątrz p, a Y' może znajdować się gdziekolwiek w drzewie). Jeśli końcówka Y nie jest niczemu przypisana, to nie ma problemu, po prostu przepinamy. Przypuśćmy, że końcówka Y jest połączona z pewnym pokojem X' (rys. 1). Istotne jest, że połączenia między X' i Y oraz między X i Y' przechodzą przez korzeń p, gdyż pokoje X i Y znajdują się w obrębie p, a w obrębie poddrzew p występują jedynie połączenia zgodne z wymaganiami naszego algorytmu, podczas gdy te takie nie są. Zamieniamy te dwa połączenia: łączymy X z Y oraz X' z Y'. Aby móc to zrobić, musimy mieć pewność, że nowe ścieżki są nie dłuższe niż k. Ścieżka z X do Y ma długość k. Odległość od korzenia p do Y' to co najwyżej $k-i$ (bo istniało połączenie z X do Y'), a odległość od korzenia p do X' to co najwyżej i (bo istniało połączenie z X' do Y) — a zatem rzeczywiście ścieżka z X' do Y' ma długość co najwyżej k.



Rys. 1: Zamiast łączyć X z Y' i X' z Y, można połączyć X z Y i X' z Y'.

- Mamy pokój X na głębokości i, który zgodnie z algorytmem ma być przypisany końcówce Y wystającej o $i+1$ w górę (czyli gaśnicy na głębokości $k-i-1$), a teraz jest przypisany pewnej innej końcówce Y'. Jeśli końcówka Y nie jest niczemu przypisana, to możemy po prostu przypisać X do Y. Przypuśćmy, że końcówka Y jest połączona z pewnym pokojem X'. Połączenia między X' i Y oraz między X i Y' przechodzą przez korzeń p, jak poprzednio. Wiemy, że odległość od korzenia p do Y' to co najwyżej $k-i$ (bo istniało połączenie z X do Y'), a odległość od korzenia p do X' to co najwyżej $i+1$ (bo istniało połączenie z X' do Y). Jeśli przynajmniej jedna z tych nierówności jest ostra, to możemy połączyć X z Y oraz X' z Y' i obie ścieżki będą długości co najwyżej k. Co jednak, jeśli w obu miejscach mamy równości? Zauważmy, że wtedy oba pokoje X' i Y' znajdują się poza p. Gdyby bowiem na przykład pokój X' był wewnątrz p, to nasz algorytm mógłby połączyć X' z Y ścieżką długości k, co ma wyższy priorytet (krok 4) niż łączenie X z Y ścieżką długości $k-1$ (krok 5). Podobnie, gdyby końcówka Y' była wewnątrz p, to nasz algorytm raczej połączyłby X z Y' ścieżką długości k niż X z Y ścieżką długości $k-1$. Skoro więc oba X' i Y' są poza p, to najkrótsza ścieżka między nimi nie przechodzi przez korzeń p, lecz jest co najmniej o 2 krótsza niż

  (odległość od korzenia p do X') + (odległość od korzenia p do Y'),

  czyli jest długości co najwyżej $k-1$. Zatem możemy dokonać zamiany połączeń.

W końcu dochodzimy do korzenia. Zauważmy, że nasz algorytm w korzeniu maksymalizuje liczbę połączeń istniejących końcówek z pokojami nieprzypisanymi jeszcze do żadnej gaśnicy — szerszą argumentację uzasadniającą ten fakt pozostawiamy już Czytelnikowi. Wynika z tego, że w każdym innym rozwiązaniu potrzeba w korzeniu co najwyżej tyle gaśnic, ile w naszym algorytmie. Ponadto, każdy inny sposób połączenia można sprowadzić w oczywisty sposób do uzyskanego przez nasz algorytm (po prostu zapominamy o starych połączeniach i łączymy tak, jak w naszym algorytmie; tym razem nie musimy się martwić o to, co dzieje się poza aktualnym poddrzewem, bo rozważamy już całe drzewo). Tym oto sposobem dobrnęliśmy do końca dowodu poprawności naszego rozwiązania.

Opisane powyżej rozwiązanie zostało zaimplementowane w plikach `gas.cpp`, `gas1.pas` oraz `gas2.java`.

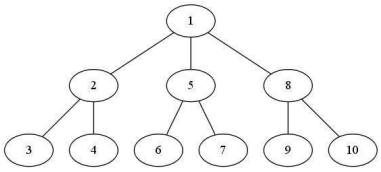
Testy

Rozwiązania zadania były sprawdzane na 10 zestawach danych testowych, w każdym zestawie zgrupowane były dwa lub trzy testy. Począwszy od piątej grupy, testy z końcówką *a* to testy wydajnościowe. Testy 10*b*, 10*c* sprawdzają użycie typu całkowitego 64-bitowego w rozwiązaniu wzorcowym. Wszystkie testy zostały wygenerowane losowo, jednak dla różnych parametrów charakteryzujących wygląd drzewa. W poniższej tabelce przedstawiamy statystyki poszczególnych testów. Kolumny n, s, k określają parametry z treści zadania, d_{max} to maksymalny stopień (liczba sąsiadów) wierzchołka w drzewie, *diam* to średnica drzewa (maksymalna odległość między dwoma wierzchołkami), natomiast g to potrzebna liczba gaśnic (wynik).

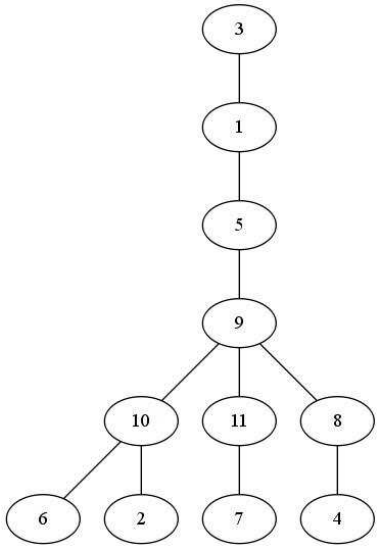
Nazwa	**n**	**s**	**k**	d_{max}	*diam*	g	**Opis**
gas1a.in	10	10	3	3	4	1	
gas1b.in	11	2	2	5	5	6	
gas2a.in	20	20	1	18	3	2	duże d_{max}
gas2b.in	25	3	5	4	9	9	
gas3a.in	159	10	3	7	8	16	
gas3b.in	400	200	20	11	15	2	duże s
gas4a.in	1 000	7	4	8	18	143	
gas4b.in	2 000	9	18	6	122	223	dość duży *diam*
gas5a.in	4 728	3	20	11	12	1 576	
gas5b.in	3 702	10	16	3	186	371	drzewo binarne
gas6a.in	8 004	7	19	21	13	1 144	
gas6b.in	20 000	126	7	23	147	1 206	
gas7a.in	44 192	17	20	101	14	2 600	
gas7b.in	50 000	10 000	2	9	435	10 196	$k = 2$, więc duże g
gas8a.in	47 231	31	20	21	10	1 524	

Nazwa	n	s	k	d_{max}	*diam*	*g*	Opis
gas8b.in	70000	40	13	4	434	2286	
gas9a.in	90000	131	18	31	21	688	
gas9b.in	100000	6000	20	6	139	541	
gas10a.in	68929	4	20	3	32	17233	małe *s*
gas10b.in	90001	90001	2	30000	6	30000	duże d_{max} i *g*
gas10c.in	99001	99001	2	33000	6	33000	duże d_{max} i *g*

Poniżej zamieszczamy ilustracje dwóch pierwszych testów.



Test *gas1a.in*.



Test *gas1b.in*.