Trening (czyni Mistrza)

Limit pamięci: 256 MB

Bajtazar jest trenerem algorytmiki. Wpadł ostatnio na wspaniały pomysł – aby każdy uczestnik treningu zmierzył się z każdym innym w pojedynku implementacji Dijkstry na czas. Najlepiej gdyby udało się zrobić to tak, by żadna para pojedynkujących się nie została powtórzona! Niestety, gdy Bajtazar spróbował taki trening zorganizować, coś się pomieszało i nie dość, że niektóre pary się powtórzyły, to co gorsza nie każdy zmierzył się z każdym... Jako zdeterminowany algorytmik, postanowiłeś wziąć rozplanowanie sposobu doboru par zawodników na kolejnym treningu w swoje ręce.

Na sali znajduje się n stanowisk komputerowych ponumerowanych od 1 do n. Uczestnicy ustawiają się w pary w taki sposób, że:

- Osoba na stanowisku 1 pojedynkuje się z osobą na stanowisku 2.
- Osoba na stanowisku 3 pojedynkuje się z osobą na stanowisku 4.
- ...osoba na stanowisku $2k-1$ pojedynkuje się z osobą na stanowisku $2k$.

(Wygodnie myśleć o tym jako o dwóch rzędach stanowisk komputerowych – pierwszy rząd jest numerowany liczbami nieparzystymi, drugi rząd jest numerowany liczbami parzystymi – pojedynkujący się mogą spojrzeć sobie nad ekranami laptopów głęboko w oczy).

Jeżeli liczba osób n jest nieparzysta, to osoba znajdująca się na pozycji n nie ma pary. Jest to idealny moment na podglądanie implementacji kolegów i krytykowanie sposobu stawiania przez nich klamer czy wielkości wcięcia. Turniej będzie odbywać się w rundach. Runda wygląda tak:

1. Każda para (przypominamy – parę tworzą osoba na pozycji $2i-1$ z osobą na pozycji $2i$) rozpoczyna pojedynek w *klepaniu Dijkstry na czas*.
2. Trener woła **Przekroczono limit czasu! Przejście!** ...i wszyscy uczestnicy przechodzą na nowe pozycje zgodnie z ustaloną przez Ciebie *regułą przejścia*.
3. Po przejściu tworzą się nowe pary (zgodnie z nowymi pozycjami) i zaczyna się kolejna runda turnieju.

Twoim zadaniem jest zaplanowanie serii przejść tak, aby na koniec treningu okazało się, że każda dwójka uczestników zmierzyła się ze sobą. Ponadto Bajtazar będzie bardzo szczęśliwy, jeżeli żadna para się nie powtórzy. Jeżeli już jednak jakieś pary mają się powtórzyć, to nie przesadzaj chociaż z liczbą rund – zadowolenie Bajtazara będzie zależne od rozwlekłości twojego planu (sekcja **Ocenianie**) – niestety mimo najlepszych chęci Bajtazara oraz uczestników, **contest** nie może trwać wiecznie...

Formalnie – początkowo i-ta osoba zajmuje stanowisko i. Regułą przejścia nazywamy ciąg liczb $p_1, p_2, \ldots, p_n$ (permutację liczb od 1 do n). Oznacza on, że osoba, która **aktualnie** zajmuje stanowisko i, ma przejść na stanowisko o numerze p_i. Po każdym przejściu przy każdym stanowisku ma się znajdować dokładnie jedna osoba.

Twoim zadaniem jest wypisanie kolejnych reguł przejść, które pozwolą jak najlepiej przeprowadzić zajęcia.

Wejście

W pierwszym wierszu wejścia znajduje się jedna liczba naturalna n $(3 \leq n \leq 100)$ – liczba osób na treningu.

Wyjście

W pierwszym wierszu wypisz liczbę przejść k $(0 \leq k \leq n^2)$. W kolejnych k wierszach wypisz opisy kolejnych reguł przejść. Każdy opis to n liczb: $p_1, p_2, \ldots, p_n$, gdzie p_i oznacza **numer stanowiska, na które ma przejść osoba zajmująca aktualnie stanowisko o numerze** i. Każda kolejna wypisana przez Ciebie reguła może być inna, ale nie musi – reguły mogą się powtarzać. Reguły przejścia będą aplikowane w wypisanej przez Twój program kolejności.

Trening (czyni Mistrza)

Limit pamięci: 256 MB

Przykład

Wejście dla testu `r3c0`:

```
3
```

Wyjście dla testu `r3c0`:

```
2
3 1 2
3 1 2
```

Okazuje się, że dla $n = 3$ wystarczy jedna prosta reguła zaaplikowana dwa razy. W pierwszej rundzie (przed pierwszym przejściem) pojedynkuje się para (1, 2), a trzecia osoba czeka. Po pierwszym przejściu w parze będą (2, 3). Po ostatnim przejściu utworzy się ostatnia para (1, 3) - czyli każda para zawodników zostanie uwzględniona. Jak myślisz, czy dla większych n też będzie tak łatwo???

Ocenianie

Rozwiązanie zostanie ocenione następująco:

- **0%**, jeśli wystąpi co najmniej jedno z: (1) po wykonaniu wszystkich przejść istnieją dwie osoby, które nigdy się ze sobą nie pojedynkowały, (2) wypiszesz niepoprawną permutację, (3) liczba przejść przekracza n^2.
- **100%**, jeśli każda para zmierzyła się ze sobą **dokładnie raz**.
- **80%**, jeśli każda para zmierzyła się ze sobą co najmniej raz (mogły być powtórki), a liczba przejść wynosi co najwyżej $2n$.
- **30%**, jeśli każda para zmierzyła się ze sobą co najmniej raz, a liczba przejść wynosi co najwyżej n^2.

Podzadanie	Ograniczenia	Limit czasu	Punkty
1	$n \leq 4$	1 s (C++) / 6 s (Python)	5
2	$n \leq 5$	1 s (C++) / 6 s (Python)	3
3	$n \leq 6$	1 s (C++) / 6 s (Python)	7
4	$n \leq 8$	1 s (C++) / 6 s (Python)	10
5	n jest nieparzyste	1 s (C++) / 6 s (Python)	35
6	Brak dodatkowych ograniczeń	1 s (C++) / 6 s (Python)	40

Eksperymenty

W sekcji **Pliki i testy** znajduje się program do wizualizacji rozwiązań (wraz z opisem sposobu użytkowania). Program symuluje trening przeprowadzony z użyciem Twoich reguł przejść. Wypisuje przy tym pomocne informacje takie jak aktualnie uformowane pary, końcową liczbę powtórek oraz liczbę nieobsłużonych par. **Gorąco zachęcam** do wypróbowania – z pewnością pozwoli to łatwiej przyswoić treść, tudzież w przyjemniejszy sposób znaleźć potencjalne błędy w niepoprawnym rozwiązaniu.

Oczywiście program ten można na swoje potrzeby dowolnie modyfikować – może to być dobra i edukacyjna zabawa (;