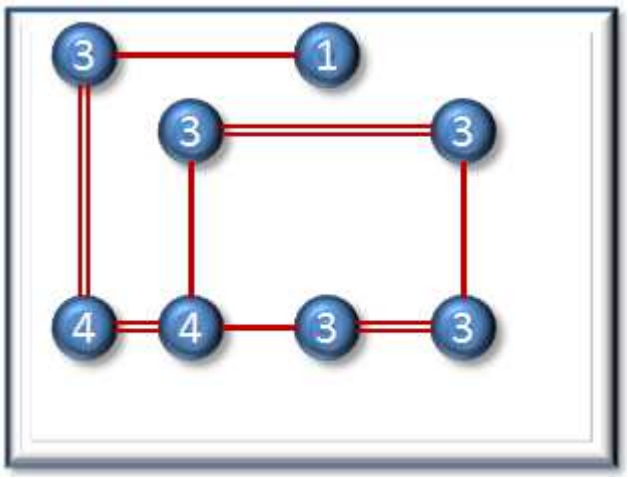
DEEC
DEPARTAMENTO DE ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES
TÉCNICO LISBOA

Mestrado Integrado em Engenharia Electrotécnica e de Computadores (MEEC)

Algoritmos e Estruturas de Dados

Enunciado do Projecto



Bridges

Versão 2.0 (27/Março/2016) – eliminação de gralhas detectadas na Secção 2.3 e Apêndice A

2015/2016
2º Semestre

Conteúdo

1 Introdução

Neste projecto pretende-se desenvolver um programa que seja capaz de construir pontes entre ilhas. Cada ilha possui a indicação de quantas pontes terá que possuir e as ilhas podem ser ligadas entre si com pontes desde que as suas coordenadas correspondam à mesma linha ou à mesma coluna e a ponte a construir não cruze outra ponte, e todas as pontes construídas comecem e acabem numa ilha. Ou seja, não podem ser construídas pontes em diagonal, nem sobrepostas, assim como nenhuma ponte pode terminar na água. Adicionalmente, um par de ilhas pode ser ligado por um máximo de duas pontes paralelas. Assim sendo, o número máximo de pontes que podem ser pedidas para uma ilha é oito. Naturalmente, uma ilha poderá necessitar, no mínimo, apenas uma ponte. Isto é, nunca serão indicadas ilhas que sejam a origem/destino de zero pontes.

Na Figura 1 apresenta-se um exemplo de um mapa de ilhas com a indicação do número de pontes para cada uma delas.

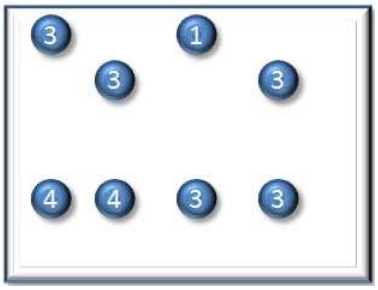


Figura 1: Exemplo de um mapa de ilhas.

De acordo com a figura, pode assumir-se que todas as ilhas estejam circunscritas num espaço rectangular de dimensões $N \times M$, contendo I ilhas, e cada ilha é definida pelas coordenadas horizontal e vertical nesse espaço rectangular. Dado um qualquer mapa, o objectivo é colocar as pontes que satisfaçam todas as restrições, podendo acontecer que um dado mapa não possua solução. Para o problema da Figura 1 a solução é a indicada na Figura 2.

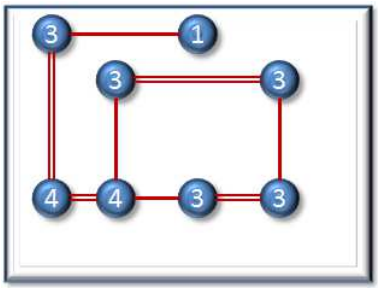


Figura 2: Exemplo de solução para um mapa de ilhas.

Deverá ser fácil confirmar que a solução apresentada cumpre as especificações do problema para o mapa da Figura 1. Para além disso, a solução da Figura 2 é única. Por outro lado, se o mapa original não possuísse a ilha indicada no topo que necessita uma ponte, então esse outro mapa não possuiria solução. Existem mapas que poderão

admitir mais que uma solução. Por exemplo, na Figura 3 apresenta-se um mapa e duas das soluções possíveis para esse mapa.

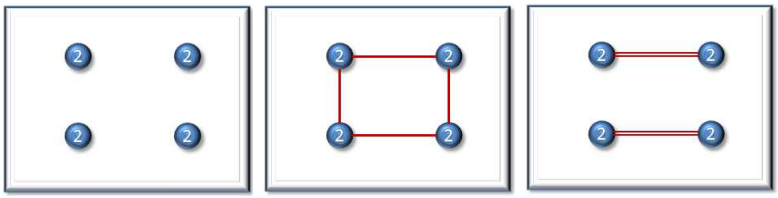


Figura 3: Exemplo de um mapa de ilhas que admite diferentes soluções.

A variante original deste problema impõe que uma solução válida tenha que garantir que todas as ilhas estejam ligadas. Ou seja, que partindo de qualquer das ilhas seja possível chegar a qualquer uma das outras ilhas. Nesse caso, e considerando de novo a Figura 3, apenas a solução central seria válida para o mapa da esquerda. No caso do projecto de AED para este semestre decidiu-se incluir esta restrição em formato opcional, como será detalhado no resto deste documento.

O que se pretende neste projecto é desenvolver uma aplicação em linguagem C que resolva mapas de forma automática em três variantes diferentes.

Nas secções seguintes descreve-se o formato de utilização do programa a desenvolver, no que diz respeito aos ficheiros de entrada e saída; as regras de avaliação; e faz-se referência ao *Código de Conduta Académica*, que se espera seja zelosamente cumprido por todos os alunos que se submetam a esta avaliação.

Aconselha-se todos os alunos a lerem com a maior atenção todos os aspectos aqui descritos. Será obrigatória a adesão sem variações nem tonalidades a todas as especificações aqui descritas. A falha em cumprir alguma(s) especificação(ões) e/ou regra(s) constante(s) neste enunciado acarretará necessariamente alguma forma de desvalorização do trabalho apresentado.

Por esta razão, tão cedo quanto possível e para evitar contratempos tardios, deverão os alunos esclarecer com o corpo docente qualquer aspecto que esteja menos claro neste texto, ou qualquer dúvida que surja após uma leitura atenta e cuidada deste enunciado.

2 O programa BRIDGES

O programa a desenvolver deverá ser capaz de ler e armazenar mapas e produzir soluções para esses mapas ou indicar não haver solução, quando esse seja o caso.

2.1 Execução do programa

O programa de BRIDGES deverá ser invocado na linha de comandos da seguinte forma:

```
aed$ ./bridges <nome1>.map
```

`bridges` designa o nome do ficheiro executável contendo o programa BRIDGES;

`<nome1>.map` em que `<nome1>` é variável, identificando o ficheiro contendo um ou mais mapas para resolver.

Para cada mapa o programa deverá fazer uma de três coisas:

Variante 1 – produzir uma qualquer solução, se existir, ou indicar que não existe solução;

Variante 2 – produzir uma qualquer solução, se existir, indicando se as ilhas nessa solução estão ou não todas ligadas, ou indicar que não existe solução;

Variante 3 – produzir uma solução obrigatoriamente com todas as ilhas ligadas, ou indicar que não existe esse tipo de solução.

2.2 Formato de entrada

O ficheiro dos mapas, poderá conter mais que um mapa. Para cada mapa contém uma ilha por linha, depois de indicar quais as dimensões que circunscrevem todas as ilhas, quantas ilhas existem no mapa e qual a variante a usar. Após a listagem de todas as ilhas, haverá sempre uma linha contendo o número -1, indicando o final da informação relativa ao mapa. Por exemplo, um ficheiro de mapas poderá ser

```
10 25 20 1
1 3 2 6
2 3 10 2
...
-1
25 10 15 3
1 12 5 2
2 1 6 3
...
-1
...
```

A informação de cada ilha será sempre feita por intermédio de quatro inteiros. O primeiro inteiro indica o nome da ilha, que estarão sempre numeradas por ordem crescente, a partir de 1. Os dois inteiros seguintes indicam as coordenadas, linha e coluna, da ilha. A linha de uma dada ilha será sempre um número entre 1 e N e a coluna será sempre um número entre 1 e M. A posição $[1, 1]$ corresponde ao canto superior esquerdo do rectângulo e a posição $[N, M]$ corresponde ao canto inferior direito. O último inteiro indica quantas pontes terão início/fim nessa ilha.

Concretamente, no exemplo acima o primeiro mapa contém 20 ilhas circunscritas num espaço de 10 linhas e 25 colunas e pretende-se usar a variante 1. Ainda para esse primeiro mapa, a ilha de nome 1 está na linha 3 e coluna 2 e deverá ter 6 pontes. A ilha de nome 2, está na linha 3 e coluna 10 e deverá ter 2 pontes. Após a indicação das ilhas do primeiro mapa, surge um segundo mapa com 15 ilhas circunscritas a um espaço com 25 linhas e 10 colunas, para o qual se deverá usar a variante 3.

Os ficheiros com um ou mais mapas poderão ter qualquer nome, mas têm obrigatoriamente a extensão `.map`.

Assume-se que todos os ficheiros de extensão `.map` estão correctos e no formato especificado anteriormente. Por essa razão, o programa não necessita fazer qualquer verificação

da sua correcção. Apenas necessita garantir que as extensões estão correctas e que esses ficheiros passados como argumento existem de facto.

2.3 Formato de saída

O resultado da execução do programa BRIDGES consiste em listar as pontes colocadas entre pares de ilhas para cada um dos mapas. Para qualquer mapa, a primeira linha da solução deverá sempre possuir quatro inteiros: o número de linhas, o número de colunas, o número de ilhas do mapa e um quarto número que será 0, 1 ou 2. Este quarto número contém informação sobre a solução obtida e o seu valor também depende da variante usada.

- Para a variante 1, qualquer que seja o resultado (com ou sem solução) este quarto número deverá ser sempre 0;
- Para a variante 2, o quarto número deverá ser 1 se não existir solução ou se as ilhas não estiverem todas ligadas;
- Para a variante 2, caso exista solução e todas as ilhas estejam ligadas, o quarto número deverá ser 2.
- Para a variante 3, o quarto número é usado da mesma forma que para a variante 2.

Após esta primeira linha, e se houver solução, cada uma das linhas seguintes diz respeito a um par de ilhas que tenham sido ligadas. Cada linha possuirá três inteiros: os primeiros dois identificam cada uma das ilhas ligadas; e o terceiro indica quantas pontes as unem: ou 1 ou 2. Como as pontes são bidireccionais, cada par ligado deverá ser indicado uma só vez e a ordem pela qual as duas ilhas ligadas é apresentada é irrelevante.

No final da indicação de todas as pontes, deverá haver uma linha com -1, indicando que a solução do mapa foi concluída. Para mapas sem solução, após a primeira linha indicativa do início de solução para um mapa, deverá existir apenas uma linha com -1.

Em síntese, as pontes colocadas em qualquer mapa surgem entre a primeira linha, com quatro inteiros, e a última linha com -1. Quando não há solução que cumpra as especificações, a primeira e a última linha da solução são adjacentes.

Se o ficheiro de mapas possuir mais que um problema, o ficheiro de saída deverá conter uma solução para cada um dos problemas indicados e pela mesma ordem em que surgem no ficheiro de entrada.

A(s) solução(ões) deve(m) ser colocada(s) num ficheiro de saída, cujo nome deve ser o mesmo do ficheiro de mapas mas **com extensão** `.sol`. Este ficheiro deve ser criado e aberto pelo programa. Por exemplo, se o ficheiro com mapas se chama `mapa231.map`, o ficheiro de saída deve-se chamar `mapa231.sol`.

Considere, por exemplo, o seguinte ficheiro de entrada:

```
4 4 8 1
1 1 1 3
2 4 1 4
3 1 3 1
4 2 2 3
5 2 4 3
6 4 2 4
7 4 3 3
8 4 4 3
-1
2 2 4 2
1 1 1 2
2 1 2 2
3 2 1 2
4 2 2 2
-1
2 2 4 3
1 1 1 2
2 1 2 2
3 2 1 2
4 2 2 2
-1
```

Este ficheiro contém três mapas. O primeiro corresponde ao mapa da Figura 1 e pede-se a variante 1. O segundo mapa corresponde ao da Figura 3 e pede-se a variante 2. O terceiro mapa é de novo o da Figura 3 e pede-se a variante 3.

A solução do primeiro mapa é a seguinte:

```
4 4 8 0
1 3 1
1 2 2
2 6 2
6 4 1
6 7 1
5 4 2
8 5 1
7 8 2
-1
```

Uma solução possível na variante 2 para o segundo mapa é:

```
2 2 4 1
1 3 2
2 4 2
-1
```

Para o terceiro mapa a solução é única e pode ser apresentada da seguinte forma:

```
2 2 4 2
1 2 1
2 4 1
4 3 1
3 1 1
-1
```

O ficheiro de saída seria a concatenação das três soluções acima descritas (ver Apêndice A).

Se o programa for invocado com um ficheiro inexistente, que não possua a extensão `.map` ou sem qualquer argumento, deverá sair silenciosamente. Ou seja, sem escrever qualquer mensagem de erro, nem criar qualquer ficheiro de saída.

3 Avaliação do Projecto

O projecto está dimensionado para ser feito por grupos de dois alunos, não se aceitando grupos de dimensão superior. Excepcionalmente poderão ser admitidas submissões de trabalhos realizados por apenas um aluno. A admissibilidade destas excepções será feita com base na demonstração de incapacidade absoluta em se ter encontrado colega de trabalho. No entanto, convém sublinhar que a avaliação dos trabalhos submetidos não merecerá qualquer espécie de bonificação quando realizados por apenas um aluno. Ou seja, todos os trabalhos submetidos serão avaliados da mesma forma.

Para os alunos que frequentam o laboratório, o grupo de projecto não tem de ser o mesmo do laboratório, mas é aconselhavel que assim seja.

Do ponto de vista do planeamento do projecto, os alunos deverão ter em consideração que o tempo de execução e a memória usada serão tidos em conta na avaliação do projecto submetido. Um dos aspectos cruciais deste projecto é o tempo de execução.

Serão admissíveis para avaliação versões do programa que não possuam todas as funcionalidades. Naturalmente que um menor número de funcionalidades operacionais acarretará penalização na avaliação final.

Quando os grupos de projecto estiverem constituídos, os alunos devem obrigatoriamente inscrever-se no sistema Fenix, no grupo de Projecto correspondente, que será criado oportunamente.

A avaliação do projecto decorre em três ou quatro instantes distintos. O primeiro instante coincide com a submissão electrónica do código do projecto. O segundo instante corresponde à entrega do relatório final em mão aos docentes, entrega essa que ratifica e lacra a submissão electrónica anteriormente realizada. É possível aos alunos submeter o projecto e entregar o relatório durante três dias consecutivos. No entanto, entregas depois da primeira data sofrerão uma penalização (veja a Tabela 1).

Num terceiro instante há uma proposta enviada pelo corpo docente que pode conter a indicação de convocatória para a discussão e defesa do trabalho ou uma proposta de nota para a componente de projecto. Caso os alunos aceitem a nota proposta pelo docente avaliador, a discussão não é necessária e a nota torna-se final. Se, pelo contrário, os alunos decidirem recorrer da nota proposta, será marcada uma discussão de recurso em data posterior. O quarto instante acontece apenas caso haja marcação de uma discussão. Nestas circunstâncias, a discussão é obrigatoriamente feita a todo o grupo, sem prejuízo de as classificações dos elementos do grupo poderem vir a ser distintas. As datas de entrega referentes aos vários passos da avaliação do projecto estão indicadas na Tabela 1.

Tabela 1: Datas importantes do Projecto

Data	Documentos a Entregar
até 28 de Março de 2016, 2ª feira	Enunciado do projecto disponibilizado na página da disciplina.
até 6 de Maio de 2015 (24h)	Inscrição dos grupos no sistema Fenix.
18 de Maio de 2016, 4ª feira 12h 15h	**1ª Data de entrega do projecto:** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
19 de Maio de 2016, 5ª feira 12h 15h	**2ª Data de entrega do projecto: penalização de um (1) valor** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
20 de Maio de 2016, 6ª feira 12h 15h	**3ª Data de entrega do projecto: penalização de dois (2) valores** Submissão electrónica do projecto. Entrega do relatório do projecto em papel.
	Submissões posteriores a 20 de Maio têm penalização de 20 valores.
23 a 27 de Maio de 2015	Discussão do trabalho (data combinada com cada grupo).

A submissão electrónica estará disponível em data e condições a indicar posteriormente na página da disciplina e serão aceites trabalhos entregues até ao instante final indicado. Ao contrário de anos anteriores, neste semestre **não haverá qualquer extensão nos prazos de entrega**, pelo que os alunos devem organizar o seu tempo de forma a estarem em condições de entregar a versão final na primeira data indicada. As restantes datas devem ser encaradas como soluções de recurso, para a eventualidade de alguma coisa correr menos bem durante o processo de submissão. O relatório final deverá ser entregue em mão aos docentes no laboratório no dia indicado na Tabela 1.

Note-se que o projecto só é considerado entregue aquando da entrega do relatório em papel. As submissões electrónicas do código não são suficientes para concretizar a entrega. Um grupo que faça a submissão electrónica do código e a entrega do relatório em papel, por exemplo, na 1ª data de entrega pode fazer submissões nas datas seguintes, mas se fizer a entrega de um novo relatório em papel, será este, e as respectivas submissões, o considerado para avaliação, com a penalização indicada.

3.1 Funcionamento

A verificação do funcionamento do código a desenvolver no âmbito do projecto será exclusivamente efectuada nas máquinas do laboratório da disciplina, embora o desenvolvimento possa ser efectuado em qualquer plataforma ou sistema que os alunos escolham. Esta regra será estritamente seguida, não se aceitando quaisquer excepções. Por esta razão, é essencial que os alunos, independentemente do local e ambiente em que desenvolvam os seus trabalhos, os verifiquem no laboratório antes de os submeterem, de forma a evitar

problemas de última hora. Uma vez que os laboratórios estão abertos e disponíveis para os alunos em largos períodos fora do horário das aulas, este facto não deverá causar qualquer tipo de problemas.

3.2 Código

Não deve ser entregue código em papel. Os alunos devem entregar por via electrónica o código do programa (ficheiros `.h` e `.c`) e uma `Makefile` para gerar o executável. Todos os ficheiros (`*.c`, `*.h` e `Makefile`) devem estar localizados na directoria raiz.

O código deve ser estruturado de forma lógica em vários ficheiros (`*.c` e `*.h`). As funções devem ter um cabeçalho curto mas explicativo e o código deve estar correctamente indentado e com comentários que facilitem a sua legibilidade.

3.3 Relatório

Os relatórios devem ser entregues na altura indicada na Tabela 1. O relatório do projecto deverá contemplar os aspectos seguidamente indicados. A apresentação do mesmo deverá iniciar-se por aspectos de ordem geral, seguindo-se descrições mais detalhadas dos módulos e estruturas de dados utilizados. Sugere-se a seguinte estrutura para o relatório:

- Uma capa com os dados dos membros do grupo, incluindo nome, número e e-mail. Esta capa deverá seguir o formato indicado na página da disciplina (oportunamente será disponibilizado);
- Uma página com o índice das secções em que o relatório se divide;
- Uma descrição do problema que foi resolvido, com indicação clara das especificações do mesmo tal como foram entendidas;
- Um texto simples que indique como os alunos abordaram e resolveram o problema;
- Uma descrição completa da arquitectura do programa, incluindo fluxogramas detalhados e um texto claro, mas sucinto, indicando a divisão lógica e funcional dos módulos desenvolvidos para a resolução do problema, explicitando os respectivos objectivos, as funções utilizadas e as estruturas de dados de suporte;
- Uma descrição detalhada das tipos de dados utilizadas e justificação das mesmas (tabelas, listas, filas, pilhas, árvores, grafos, acervos, etc.);
- Descrição dos algoritmos usados (por exemplo, na manipulação dos tipos de dados);
- Uma descrição dos subsistemas funcionais que existam e, para cada um
  - a descrição dos objectivos do subsistema (até 5 linhas);
  - o nome do módulo onde estão definidos os tipos de dados abstractos (ficheiros `.h`) a utilizar no subsistema;
  - o nome do módulo `C` onde estão as funções do respectivo subsistema;
  - listagem das funções implementadas no subsistema, indicando para cada uma a respectiva assinatura e os objectivos da função (descrição sumária, sem código);
- Uma análise dos requisitos computacionais do programa desenvolvido, tanto em termos da memória que utiliza como da complexidade computacional, com particular ênfase no custo das operações de processamento sobre os tipos de dados usados e/ou criados;

- Ainda sobre a análise computacional, os grupos deverão ter em consideração que, para este problema, existem três parâmetros de interesse: N, M e I. A análise computacional deverá ser capaz de indicar qual a influência de cada um destes três parâmetros. O número de pontes a colocar também poderá ter influência na complexidade do programa;
- Uma análise crítica do funcionamento do programa e a avaliação do desempenho do projecto implementado;
- Pelo menos, um pequeno exemplo completo e detalhado de aplicação, com descrição da utilização das estruturas de dados em cada passo e de como são tomadas as decisões.

3.4 Critérios de Avaliação

Os projectos submetidos serão avaliados de acordo com a seguinte grelha:

- Testes passados na submissão electrónica – 75%
  Para a variante 1 haverá 15 a 16 testes, para a variante 2 haverá 2 a 3 testes e para a variante 3 haverá um máximo de 2 testes.
- Estruturação do código e comentários – 5%
- Gestão de memória e tipos abstractos – 5%
- Relatório escrito – 15%

Na submissão electrónica, cada projeto será testado com 20 ficheiros de mapas de diferentes graus de complexidade, onde se avaliará a capacidade de produzir soluções correctas dentro de limites de tempo e memória. Para o limite de tempo, cada um dos testes terá de ser resolvido em menos de 60 segundos. Para o limite de memória, cada um dos testes não poderá exceder 100MB como pico de memória usada. Cada teste resolvido dentro dos orçamentos temporal e de memória que produza soluções correctas recebe um ponto.

Se o corpo docente entender necessário, face à complexidade dos problemas a resolver, poderão os limites de tempo e/ou memória ser aumentados.

Caso o desempenho de alguma submissão electrónica não seja suficientemente conclusivo, poderá ser sujeita a testes adicionais fora do contexto da submissão electrónica. Alguns desses testes adicionais poderão ser os mesmos da submissão electrónica, que serão corridos sem limite de tempo nem de memória. Outros poderão ser diversos daqueles. O desempenho nesses testes adicionais poderá contribuir para subir ou baixar a pontuação obtida na submissão electrónica.

No que à avaliação do relatório diz respeito, os elementos de avaliação incluem: apreciação da abordagem geral ao problema e respectiva implementação; análise de complexidade temporal e de memória; exemplo de aplicação; clareza e suficiência do texto, na sua capacidade de descrever e justificar com precisão o que está feito; e qualidade do texto escrito e estruturação do relatório.

Pela análise da grelha de avaliação aqui descrita, deverá ficar claro que a ênfase da avaliação se coloca na capacidade de um programa resolver correctamente os problemas a que for submetido. Ou seja, o código de uma submissão até pode ser muito bonito e bem estruturado e o grupo até pode ter dispendido muitas horas no seu desenvolvimento. No entanto, se esse código não resolver um número substancial de testes na submissão electrónica dificilmente terá uma nota substancial.

4 Código de Honestidade Académica

Espera-se que os alunos conheçam e respeitem o Código de Honestidade Académica que rege esta disciplina e que pode ser consultado na página da cadeira. O projecto é para ser planeado e executado por grupos de dois alunos e é nessa base que será avaliado. Quaisquer associações de grupos ou outras, que eventualmente venham a ocorrer, serão obviamente interpretadas como violação do Código de Honestidade Académica e terão como consequência a anulação do projecto aos elementos envolvidos.

Lembramos igualmente que a verificação de potenciais violações a este código é feita de forma automática com recurso a sofisticados métodos de comparação de código, que envolvem não apenas a comparação directa do código mas também da estrutura do mesmo. Esta verificação é feita com recurso ao software disponibilizado em

`http://moss.stanford.edu/` .

A Ficheiro de saída

Apresenta-se aqui o ficheiro de saída completo para o ficheiro de entrada discutido na secção 2.3.

```
4 4 8 0
1 3 1
1 2 2
2 6 2
6 4 1
6 7 1
5 4 2
8 5 1
7 8 2
-1
2 2 4 1
1 3 2
2 4 2
-1
2 2 4 2
1 2 1
2 4 1
4 3 1
3 1 1
-1
```