Geometria Computacional

Cristina G. Fernandes

Departamento de Ciência da Computação do IME-USP

http://www.ime.usp.br/~cris/

segundo semestre de 2014

Interseção de segmentos

Problema: Dados n segmentos, determinar se dois deles se intersectam.

Interseção de segmentos

Problema: Dados n segmentos, determinar se dois deles se intersectam.

Aula passada: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Interseção de segmentos

Problema: Dados n segmentos, determinar se dois deles se intersectam.

Aula passada: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Estrutura de dados:

- árvore de busca binária balanceada (ABBB) ou
- skip list (tempo esperado $\mathrm{O}(n \lg n)$)

Interseção de segmentos

Problema: Dados n segmentos, determinar se dois deles se intersectam.

Aula passada: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Estrutura de dados:

- árvore de busca binária balanceada (ABBB) ou
- skip list (tempo esperado $\mathrm{O}(n \lg n)$)

Hipótese simplificadora:
Não há dois pontos extremos com mesma x-coordenada.

Em particular, não há segmentos verticais,
nem dois segmentos com extremos coincidentes.

Interseção de segmentos

Problema: Dados n segmentos, determinar se dois deles se intersectam.

Aula passada: algoritmo $\mathrm{O}(n \lg n)$ para esse problema.

Estrutura de dados:

- árvore de busca binária balanceada (ABBB) ou
- skip list (tempo esperado $\mathrm{O}(n \lg n)$)

Hipótese simplificadora:
Não há dois pontos extremos com mesma x-coordenada.

Em particular, não há segmentos verticais,
nem dois segmentos com extremos coincidentes.

Pré-processamento: ordene os extremos dos segmentos por x-coordenada.

Como evitar a hipótese simplificadora?

Pontos extremos com mesma x-coordenada:

Se existir um segmento vertical,
deixe o extremo inferior no vetor e e o superior no vetor d.

Como evitar a hipótese simplificadora?

Pontos extremos com mesma x-coordenada:

Se existir um segmento vertical,
deixe o extremo inferior no vetor e e o superior no vetor d.

Se houver extremos repetidos, há interseção.

Como evitar a hipótese simplificadora?

Pontos extremos com mesma x-coordenada:

Se existir um segmento vertical,
deixe o extremo inferior no vetor e e o superior no vetor d.

Se houver extremos repetidos, há interseção.

Extremo esquerdo de um segmento:

- extremo cuja x-coordenada é menor.
- caso o segmento seja vertical,
  chame de esquerdo o extremo
  com y-coordenada menor.

Como evitar a hipótese simplificadora?

Pontos extremos com mesma x-coordenada:

Se existir um segmento vertical,
deixe o extremo inferior no vetor e e o superior no vetor d.

Se houver extremos repetidos, há interseção.

Extremo esquerdo de um segmento:

- extremo cuja x-coordenada é menor.
- caso o segmento seja vertical,
  chame de esquerdo o extremo
  com y-coordenada menor.

O outro extremo é o direito.

Como evitar a hipótese simplificadora?

Pontos extremos com mesma x-coordenada:

Se existir um segmento vertical,
deixe o extremo inferior no vetor e e o superior no vetor d.

Se houver extremos repetidos, há interseção.

Extremo esquerdo de um segmento:

- extremo cuja x-coordenada é menor.
- caso o segmento seja vertical,
  chame de esquerdo o extremo
  com y-coordenada menor.

O outro extremo é o direito.

Extremos-Ordenados(n, S): ordena os extremos dos n segmentos em S e já dá a resposta se houver repetição.

Detecção de interseção

Detecta-Interseção(n, S)

$E \leftarrow$ Extremos-Ordenados(n, S)
$T \leftarrow \emptyset$ $\quad \triangleright$ ABBB ou skip list
para cada $p \in E$ **faça**
    $s \leftarrow segmento(p)$
    $pred \leftarrow$ Predecessor(T, s) $\quad suc \leftarrow$ Sucessor(T, s)
    se p é extremo esquerdo de s
        então Insere(T, s)
            se $(pred \neq \text{NIL}$ **e** Intersecta$(s, pred))$
            ou $(suc \neq \text{NIL}$ **e** Intersecta$(s, suc))$
                então devolva VERDADE
        senão Remove(T, s)
            se $pred$ **e** $suc \neq \text{NIL}$ **e** Intersecta$(pred, suc)$
                então devolva VERDADE
devolva FALSO

Inserção em ABB rubro-negra

InsiraRec (T, x)

se $T = \textsc{nil}$
  então $q \leftarrow$ NovaCélula$(x, \textsc{nil}, \textsc{nil},$ RUBRO$)$
      devolva q
se $x < \mathit{info}(T)$ ▷ Vamos alterar aqui!
  então $\mathit{esq}(T) \leftarrow$ InsiraRec$(\mathit{esq}(T), x)$
  senão $\mathit{dir}(T) \leftarrow$ InsiraRec$(\mathit{dir}(T), x)$
se Rubro$(\mathit{dir}(T))$ **e** Negro$(\mathit{esq}(T))$
  então $T \leftarrow$ GireEsq(T)
se Rubro$(\mathit{esq}(T))$ **e** Rubro$(\mathit{esq}(\mathit{esq}(T)))$
  então $T \leftarrow$ GireDir(T)
se Rubro$(\mathit{esq}(T))$ e Rubro$(\mathit{dir}(T))$
  então TroqueCores(T)
devolva T

Inserção em ABB rubro-negra

INSIRAREC (T, e, d, i)

se $T = \text{NIL}$
  então $q \leftarrow$ NOVACÉLULA$(i, \text{NIL}, \text{NIL},$ RUBRO$)$
    devolva q
se ESQUERDA$(e[segmento(T)], d[segmento(T)]), e[i])$
  então $esq(T) \leftarrow$ INSIRAREC$(esq(T), i)$
  senão $dir(T) \leftarrow$ INSIRAREC$(dir(T), i)$
se RUBRO$(dir(T))$ **e** NEGRO$(esq(T))$
  então $T \leftarrow$ GIREESQ(T)
se RUBRO$(esq(T))$ **e** RUBRO$(esq(esq(T)))$
  então $T \leftarrow$ GIREDIR(T)
se RUBRO$(esq(T))$ e RUBRO$(dir(T))$
  então TROQUECORES(T)
devolva T

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Você consegue projetar um algoritmo que consuma tempo $\mathrm{O}(n \lg n)$ para este problema?

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Você consegue projetar um algoritmo
que consuma tempo $\mathrm{O}(n \lg n)$ para este problema?

No máximo, quantos pares teremos que imprimir?

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Você consegue projetar um algoritmo que consuma tempo $\mathrm{O}(n \lg n)$ para este problema?

No máximo, quantos pares teremos que imprimir?

Algoritmos sensíveis à saída (*output sensitive*).

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Como adaptar o algoritmo de Shamos e Hoey?

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Como adaptar o algoritmo de Shamos e Hoey?

Novo tipo de ponto evento: as interseções.

Como tratá-las?

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Como adaptar o algoritmo de Shamos e Hoey?

Novo tipo de ponto evento: as interseções.

Como tratá-las?

Ao detectar cada uma, além de imprimi-la,
a colocamos na fila de eventos (que é agora dinâmica).

Todas as interseções de segmentos

Problema: Dada uma coleção de n segmentos no plano, encontrar todos os pares de segmentos da coleção que se intersectam.

Como adaptar o algoritmo de Shamos e Hoey?

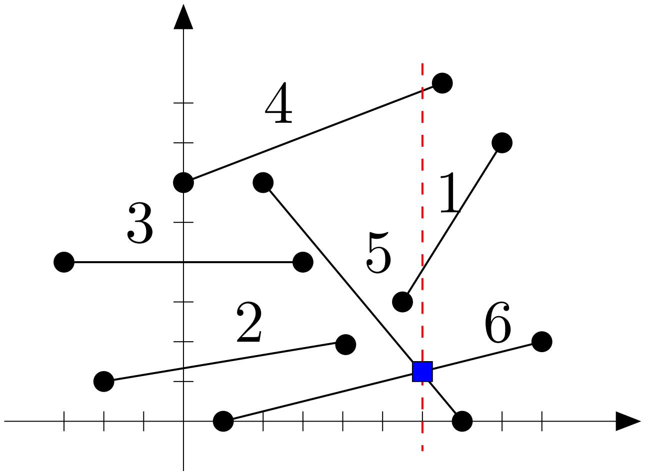
Novo tipo de ponto evento: as interseções.

Como tratá-las?

Ao detectar cada uma, além de imprimi-la,
a colocamos na fila de eventos (que é agora dinâmica).

Ao processar um ponto evento que é uma interseção, deve-se inverter a ordem dos segmentos que se intersectam neste ponto.

Ponto evento: interseção



Antes do ponto evento: $4 \prec 1 \prec 5 \prec 6$

Depois do ponto evento: $4 \prec 1 \prec 6 \prec 5$

Algoritmo de Bentley e Ottmann

Entrada: coleção $e[1\,..\,n], d[1\,..\,n]$ de segmentos.

Algoritmo de Bentley e Ottmann

Entrada: coleção $e[1\,.\,.\,n], d[1\,.\,.\,n]$ de segmentos.

Saída: todos os pares de segmentos da coleção que se intersectam.

Algoritmo de Bentley e Ottmann

Entrada: coleção $e[1\,..\,n], d[1\,..\,n]$ de segmentos.

Saída: todos os pares de segmentos da coleção que se intersectam.

Hipótese simplificadora:

Não há dois pontos eventos com a mesma x-coordenada.

Em particular, não há interseção com mesma x-coordenada que outra, ou com algum extremo de segmento.

Algoritmo de Bentley e Ottmann

Entrada: coleção $e[1\,..\,n], d[1\,..\,n]$ de segmentos.

Saída: todos os pares de segmentos da coleção que se intersectam.

Hipótese simplificadora:

Não há dois pontos eventos com a mesma x-coordenada.

Em particular, não há interseção com mesma x-coordenada que outra, ou com algum extremo de segmento.

Não há interseções múltiplas, ou seja, não há um ponto em mais do que dois segmentos da coleção.

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

ABBB com ordem dada pelas x-coordenadas dos pontos.

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

ABBB com ordem dada pelas x-coordenadas dos pontos.

A fila começa com os extremos dos intervalos.

A cada iteração,
removemos um evento da fila para processá-lo.

Fila de eventos

Agora ela é dinâmica: sofre inserções
(e, como antes, remoções).

Que ED usar para a fila de eventos?

ABBB com ordem dada pelas x-coordenadas dos pontos.

A fila começa com os extremos dos intervalos.

A cada iteração,
removemos um evento da fila para processá-lo.

Ao detertar uma interseção,
inserimos tal ponto na fila de eventos.

Quantos elementos estão na fila no pior caso?

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S.

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S.

Acha-Interseções(n, S)
1 $Q \leftarrow$ Extremos(n, S) $\triangleright$ inicializa a ABBB Q com os extremos
2 $T \leftarrow \emptyset$
3 **enquanto não** Vazia(Q) **faça**
4 $p \leftarrow$ Extrai-Min(Q)
5 Trata-Evento(p)

Versão simplificada

Hipótese simplificadora: não há pontos extremos repetidos, e as interseções são em pontos internos dos segmentos.

Extremos-Ordenados(n, S):
ordena os extremos dos n segmentos em S.

Acha-Interseções(n, S)
$Q \leftarrow$ Extremos(n, S) ▷ inicializa a ABBB Q com os extremos
$T \leftarrow \emptyset$
enquanto não Vazia(Q) **faça**
    $p \leftarrow$ Extrai-Min(Q)
    Trata-Evento(p)

Notação: Para dois pontos-evento p e q,
escrevemos $p \prec q$ se $p_x < q_x$ ou ($p_x = q_x$ e $p_y < q_y$)

Versão simplificada

Trata-Evento(p)

se p é extremo esquerdo de um segmento s
  então Insere(T, s)
    $pred \leftarrow$ Predecessor(T, s)
    $suc \leftarrow$ Sucessor(T, s)
    se $pred \neq$ NIL **e** Intersecta$(s, pred)$
      então Verifica-Novo-Evento$(p, Q, s, pred)$
    se $suc \neq$ NIL **e** Intersecta(s, suc)
      então Verifica-Novo-Evento(p, Q, s, suc)

Versão simplificada

Trata-Evento(p)

se p é extremo esquerdo de um segmento s
  então Insere(T, s)
    $pred \leftarrow$ Predecessor(T, s)
    $suc \leftarrow$ Sucessor(T, s)
    se $pred \neq$ NIL **e** Intersecta$(s, pred)$
      então Verifica-Novo-Evento$(p, Q, s, pred)$
    se $suc \neq$ NIL **e** Intersecta(s, suc)
      então Verifica-Novo-Evento(p, Q, s, suc)

Verifica-Novo-Evento(p, Q, s_1, s_2)

$q \leftarrow$ Ponto-de-Interseção(s_1, s_2)
se $q \succ p$ **e não** Pertence(Q, q)
  então Insere(Q, q)
    imprima q

Versão simplificada

Trata-Evento(p)

se p é extremo esquerdo de um segmento s
    então Insere(T, s)
        $pred \leftarrow$ Predecessor(T, s)
        $suc \leftarrow$ Sucessor(T, s)
        se $pred \neq$ NIL **e** Intersecta$(s, pred)$
            então Verifica-Novo-Evento$(p, Q, s, pred)$
        se $suc \neq$ NIL **e** Intersecta(s, suc)
            então Verifica-Novo-Evento(p, Q, s, suc)
se p é extremo direito de um segmento s
    então Remove(T, s)
        $pred \leftarrow$ Predecessor(T, s)
        $suc \leftarrow$ Sucessor(T, s)
        se $pred$ **e** $suc \neq$ NIL **e** Intersecta$(pred, suc)$
            então Verifica-Novo-Evento$(p, Q, suc, pred)$

Versão simplificada

Trata-Evento(p)
$\ldots$
se p é ponto de interseção
  então sejam s e s' os segmentos em T que contém p
    $pred \leftarrow$ Predecessor(T, s)
    $suc \leftarrow$ Sucessor(T, s')
    Remove(T, s)   Remove(T, s')
    $\triangleright$ insere s e s' na ordem inversa
    Insere(T, s')   Insere(T, s)
    se $pred \neq$ NIL **e** Intersecta($pred, s'$)
      então Verifica-Novo-Evento($p, Q, pred, s'$)
    se $suc \neq$ NIL **e** Intersecta(s, suc)
      então Verifica-Novo-Evento(p, Q, s, suc)

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

Cada iteração faz uma chamada a Predecessor, Sucessor, e uma a Insere ou Remove, na ABBB T.

Na ABBB T, em qualquer momento, há $\mathrm{O}(n)$ segmentos.

Assim, cada operação destas consome tempo $\mathrm{O}(\lg n)$.

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

Cada iteração faz uma chamada a Predecessor, Sucessor, e uma a Insere ou Remove, na ABBB T.

Na ABBB T, em qualquer momento, há $\mathrm{O}(n)$ segmentos.

Assim, cada operação destas consome tempo $\mathrm{O}(\lg n)$.

Cada iteração faz uma chamada a Extrai-Min e, eventualmente, uma a Insere na ABBB F.

Na ABBB F, em qq momento, há $\mathrm{O}(n + i) = \mathrm{O}(n^2)$ pontos.

Assim, cada operação consome tempo $\mathrm{O}(\lg n^2) = \mathrm{O}(\lg n)$.

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

Cada iteração faz uma chamada a Predecessor, Sucessor, e uma a Insere ou Remove, na ABBB T.

Na ABBB T, em qualquer momento, há $\mathrm{O}(n)$ segmentos.

Assim, cada operação destas consome tempo $\mathrm{O}(\lg n)$.

Cada iteração faz uma chamada a Extrai-Min e, eventualmente, uma a Insere na ABBB F.

Na ABBB F, em qq momento, há $\mathrm{O}(n + i) = \mathrm{O}(n^2)$ pontos.

Assim, cada operação consome tempo $\mathrm{O}(\lg n^2) = \mathrm{O}(\lg n)$.

As demais operações efetuadas em uma iteração consomem tempo $\mathrm{O}(1)$ (mesmo as chamadas a INTER).

Consumo de tempo

Seja i o número de interseções.
O algoritmo executa $2n + i$ iterações.

Cada iteração faz uma chamada a Predecessor, Sucessor, e uma a Insere ou Remove, na ABBB T.

Na ABBB T, em qualquer momento, há $\mathrm{O}(n)$ segmentos.

Assim, cada operação destas consome tempo $\mathrm{O}(\lg n)$.

Cada iteração faz uma chamada a Extrai-Min e, eventualmente, uma a Insere na ABBB F.

Na ABBB F, em qq momento, há $\mathrm{O}(n + i) = \mathrm{O}(n^2)$ pontos.

Assim, cada operação consome tempo $\mathrm{O}(\lg n^2) = \mathrm{O}(\lg n)$.

O consumo de tempo por iteração é $\mathrm{O}(\lg n)$, e o algoritmo de Bentley e Ottmann consome tempo $\mathrm{O}((n + i) \lg n)$.

Versão completa

O que fazer com os casos que excluímos?

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos-evento, sem repetições.
- Ponto-evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento de processamento do ponto.

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos-evento, sem repetições.
- Ponto-evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento de processamento do ponto.

Ao processar um ponto-evento,
determinam-se todos os segmentos que o contém
(pela lista do ponto e/ou pelos segmentos em T).

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos-evento, sem repetições.
- Ponto-evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento de processamento do ponto.

Ao processar um ponto-evento,
determinam-se todos os segmentos que o contém
(pela lista do ponto e/ou pelos segmentos em T).

Se mais de um segmento o contém, imprimimos o ponto.

Versão completa

O que fazer com os casos que excluímos?

Alterações:

- Q conterá os pontos-evento, sem repetições.
- Ponto-evento extremo: tem a lista dos segmentos que têm esse ponto como extremo.
- impressão apenas no momento de processamento do ponto.

Ao processar um ponto-evento,
determinam-se todos os segmentos que o contém
(pela lista do ponto e/ou pelos segmentos em T).

Se mais de um segmento o contém, imprimimos o ponto.

Atualiza-se T.

Atualização de T

Se o ponto-evento é um extremo, faz-se como antes:

Atualização de T

Se o ponto-evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Atualização de T

Se o ponto-evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Atualização de T

Se o ponto-evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Se o ponto-evento é uma interseção

Atualização de T

Se o ponto-evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Se o ponto-evento é uma interseção

- remove-se de T todos os segmentos que o contém no interior.
- estes são incluídos novamente na ordem inversa.

Atualização de T

Se o ponto-evento é um extremo, faz-se como antes:

- extremos esquerdos causam inclusões em T.
- extremos direitos causam remoções.

Primeiro trata-se de extremos esquerdos.

Se o ponto-evento é uma interseção

- remove-se de T todos os segmentos que o contém no interior.
- estes são incluídos novamente na ordem inversa.

Os dois casos podem acontecer ao mesmo tempo...
Isso está detalhado no livro de de Berg e outros, capítulo 2.

Comentários finais

O algoritmo pode ser ajustado para imprimir, para cada ponto de interseção, a lista de segmentos que o contém.

Comentários finais

O algoritmo pode ser ajustado para imprimir, para cada ponto de interseção, a lista de segmentos que o contém.

Consumo de tempo: $\mathrm{O}((n + i) \lg n)$
(esperado, no caso de uso de skip lists)
onde i agora é o número de segmentos impressos junto com as interseções.

Comentários finais

O algoritmo pode ser ajustado para imprimir, para cada ponto de interseção, a lista de segmentos que o contém.

Consumo de tempo: $\mathrm{O}((n+i)\lg n)$
(esperado, no caso de uso de skip lists)
onde i agora é o número de segmentos impressos junto com as interseções.

Consumo de espaço: $\mathrm{O}(n)$ para T e $\mathrm{O}(n+i)$ para Q.

Comentários finais

O algoritmo pode ser ajustado para imprimir, para cada ponto de interseção, a lista de segmentos que o contém.

Consumo de tempo: $\mathrm{O}((n+i)\lg n)$
(esperado, no caso de uso de skip lists)
onde i agora é o número de segmentos
impressos junto com as interseções.

Consumo de espaço: $\mathrm{O}(n)$ para T e $\mathrm{O}(n+i)$ para Q.

Melhora: Guarde em Q apenas os pontos de interseção de segmentos que estão consecutivos em T.
Espaço cai para $\mathrm{O}(n)$.

Comentários finais

O algoritmo pode ser ajustado para imprimir, para cada ponto de interseção, a lista de segmentos que o contém.

Consumo de tempo: $\mathrm{O}((n + i) \lg n)$
(esperado, no caso de uso de skip lists)
onde i agora é o número de segmentos
impressos junto com as interseções.

Consumo de espaço: $\mathrm{O}(n)$ para T e $\mathrm{O}(n + i)$ para Q.

Melhora: Guarde em Q apenas os pontos de interseção de segmentos que estão consecutivos em T.
Espaço cai para $\mathrm{O}(n)$.

Algoritmo de Balaban: tempo $\mathrm{O}(n \lg n + i)$ e espaço $O(n)$.