Mínimo e Máximo Ótimo

Algoritmos e Estruturas de Dados 2
2017-1
Flavio Figueiredo (http://flaviovdf.github.io)

Min Max com Número Ótimo de Operações

- Vamos ver o problema de mínimo e máximo com número ótimo de passos
- Vamos utilizar (3n/2) - 2 passos
- Não existe algoritmo melhor do que isto
- Código e passo a passo aqui
  - https://goo.gl/n5wrJq
  - https://github.com/flaviovdf/AEDS2-2017-1/blob/master/exemplos/minmax3/minmax3.c

Ideia do Algoritmo

- Iniciar o mínimo e o máximo
  - Olhando os 2 primeiros elementos do vetor
  - min e max
- Percorrer os números em pares
  - n/2 operações no laço

- Para cada par, com 1 comparação eu posso:
  - Achar o menor elemento do par (menorDoPar)
  - Achar o maior elemento do par (maiorDoPar)
- Com mais 2 comparações
  - Posso atualizar o min e o max comparando com menorDoPar e maiorDoPar

- Bastante cuidado nas inicializações
  - Vetores com elementos ímpares, não acessar elementos depois do fim do vetor

Algoritmo

- O código ao lado utiliza uma função *inicializa* para inicializar
    - i
    - min
    - max
- A mesma faz 1 comparação
- Ver links
- Removi aqui por espaço

```c
void MinMax3(int *vec, int n, int *min, int *max) {
  int i = inicializa(vec, n, min, max);
  int menorDoPar;
  int maiorDoPar;
  //sem iniciar i; já foi inicializado no inicializa
  for(; i < n - 1; i += 2) {//Observe o incremento+=2.
    if(vec[i] < vec[i+1]) {      // (n-2)/2
      menorDoPar = i;
      maiorDoPar = i+1;
    }
    else {
      menorDoPar = i+1;
      maiorDoPar = i;
    }
    if(vec[menorDoPar] < *min)   // (n-2)/2
      *min = vec[menorDoPar];
    if(vec[maiorDoPar] > *max)   // (n-2)/2
      *max = vec[maiorDoPar];
  }
}
```

[Sem Muito Formalismo] Sketch de Prova

- Existe possibilidade de obter um algoritmo MaxMin mais eficiente?

- Para responder temos de conhecer o limite inferior para
essa classe de algoritmos.

- Como? Uso de um adversário.
    - Dado um modelo de computação que expresse o comportamento do algoritmo, o oráculo informa o resultado de cada passo possível (no caso, o resultado de cada comparação).

- Para derivar o limite inferior, o adversário procura sempre fazer com que o algoritmo **trabalhe o máximo**, escolhendo como resultado da próxima comparação aquele que cause o maior trabalho possível necessário para determinar a resposta final.

[Prova] Teorema e Estratégia

- Teorema: Qualquer algoritmo para encontrar o maior eo menor elemento de um conjunto com n elementos não ordenados, $n>1$, faz pelo menos $\lceil 3n/2 - 2 \rceil$ comparações.

- Prova: A técnica utilizada define um adversário que descreve o comportamento do algoritmo utilizando:
  - um conjunto de n-tuplas (estados),
  - um conjunto de regras associadas que mostram as tuplas possíveis (estados) que um algoritmo pode assumir a partir de uma dada tupla e uma única comparação.

- Lembre-se que o computador compara elementos em pares
  - Operadores lógicos aumenta o número de comparações
  - [Exemplo] Resultado de 1 comparação par a par sendo composta com outra comparação

[Prova] Estados Possíveis

- Para o problema do maior e menor elemento, utilizamos uma 4–tupla, representada por (a, b, c, d), onde os elementos de:
  - a: nunca foram comparados;
  - b: foram vencedores e nunca perderam em comparações realizadas -> possíveis max;
  - c: foram perdedores e nunca venceram em comparações realizadas -> possíveis min;
  - d: foram vencedores e perdedores em comparações realizadas.

- Você ganha do adversário re-agrupando os elementos da seguinte forma: (0, 1, 1, n - 2).

- O algoritmo

Caminhando nos Estados

- Estados
  - a: nunca foram comparados;
  - b: foram vencedores e nunca perderam em comparações realizadas -> possíveis max;
  - c: foram perdedores e nunca venceram em comparações realizadas -> possíveis min;
  - d: foram vencedores e perdedores em comparações realizadas.

- Ao comparar elementos par a par
  - Nunca aumentamos o número de elementos em a
    - Pois fiz uma comparação
  - Nunca reduzimos o número de elementos em d
    - Se um elemento já ganhou e já perdeu ele não pode ser min nem max

Olhando Pares de Elementos

- Dado os quatro estados (a, b, c, d). Os pares podem vir de:
    - (a, a) (b, b) (c, c) (d, d)
    - (a, b) (b, c) (c, d)
    - (a, c) (b, d)
    - (a, d)

- Podemos definir um ciclo de vida de um elemento

Olhando Pares de Elementos

- Analisando cada caso:
  - (a, a) (b, b) (c, c) (d, d)
  - (a, b) (b, c) (c, d)
  - (a, c) (b, d)
  - (a, d)
- Nos casos verdes
  - A resposta nunca vai estar no estado d
    - Slide anterior
  - Em particular, é inútil comparar (d, d)
    - nunca vou comparar 2 elementos de (d,d)
    - desperdício de comparações: a resposta nunca vai estar em (d,d)

Olhando Pares de Elementos

- Analisando cada caso:
  - (a, a) (b, b) (c, c) (d, d)
  - (a, b) (b, c) (c, d)
  - (a, c) (b, d)
  - (a, d)
- Nos casos azuis
  - Sempre levo um elemento de para o estado d
    - O elemento estava em b e perdeu
      - Já ganhou no passado e agora perdeu
    - O elemento estava em c e ganhou
      - Já perdeu no passado e agora ganhou
  - Caso o estado a esteja envolvido (primeira coluna)
    - O elemento de a ou vai para b ou vai para c
      - Nunca foi comparado e agora ganhou ou perdeu
  - Caso não esteja, um elemento vai para d outro fica aonde estava

Olhando Pares de Elementos

- Analisando cada caso:
  - (a, a) (b, b) (c, c) (d, d)
  - (a, b) (b, c) (c, d)
  - (a, c) (b, d)
  - (a, d)
- Nos casos vermelhos
  - Um dos elementos sempre saem de a para b e c (um para cada)
    - Os elementos nunca foram comparados
    - Um ganhou
    - Um perdeu

Como Ganhar do Adversário

- O mesmo quer que você perca (maximizando o número de comparações)
  - O mesmo vai iniciar o jogo em algum estado
  - Escolher os pares para você comparar
- Você quer minimizar o número de comparações
- Para ganhar:
  - No início, não compare os estados verdes (simplesmente se recuse, o jogo é de comparações)
    - É um desperdício no ínicio do jogo (ver slides anteriores)
      - Pense em min e max simultaneamente. Os verdes podem ajudar apenas 1
  - Sempre que você compara os estados azuis está mais perto de ganhar o jogo
    - Ou um elemento é jogado fora (estado d) ou ele é candidato a máximo ou mínimo
    - Melhor do que o verdes que não joga ninguém fora
- A pior coisa que seu adversário pode fazer então é colocar todos em vermelho (n, 0, 0, 0)

Usando sua Estratégia

- Sempre será necessário n/2 comparações para levar os elementos de a para b ou c
  - [Definição - Slides anteriores] Cada comparação com a leva um elemento para b ou c
  - Lembrando do estado inicial:
    - Você pode simplesmente sempre comparar (a,a) removendo 2 elementos de a por vez
- Após remover todo mundo de do estado a, ficamos assim:
  (0, x, y, 0) onde x + y = n
- Sabendo disso vamos realizar n-2 comparações agora
  - Realizamos até n/2 - 1 comparações nos casos azuis (na verdade n/2 - 1 - x ou n/2 - 1 - y)
    - Em algum momento vamos ficar só com 1 elemento em b e/ou c
    - Não precisamos mais olhar para tal elemento (por isto o -1). O elemento é o min ou max.
  - Por fim:
    - Esvaziamos o estado restando com comparações (b,d) ou (c,d)
    - Novamente, não precisamos comparar o último elemento (levando ao n-2 acima)

Notas Finais

- **$n/2 + n-2 = 3n/2 - 2$**
  - $n/2$ para remover os elementos de a
  - $n - 2$ para resposta

- Se seu adversário muda o estado inicial é melhor para você
  - Ele já removeu elementos de a, menos comparações
  - Ele já colocou elementos em d, nunca serão respostas

- Com o sketch acima pode-se formalizar uma prova

- Na prova formal o adversário prover apenas a sequência de entrada
  - É intuitivo ver que para qualquer entrada iniciamos em (n, 0, 0, 0)
  - Também é comum pensar no adversário que sempre tem fazer as comparações que você escolhe
    - Aqui invertemos o contexto um pouco