## Pembahasan Jual Beli Coklat

### Deskripsi Singkat

Cari keuntungan maksimum dari membeli dan menjual sebuah coklat. Misalkan prices[i] adalah harga coklat pada hari ke-i. Soal dapat direpresentasikan menjadi cari nilai maksimum dari prices[j] - prices[i], di mana 0 <= j < i < n.

### Ide

Apabila kita menjual coklat pada hari ke-i, berapa keuntungan maksimum yang mungkin didapat? Jawaban akhir didapat dengan mengambil nilai maksimum dari keuntungan apabila menjual pada hari ke-1, atau menjual pada hari ke-2, dan seterusnya hingga hari ke-n-1.

Keuntungan maksimum dari menjual coklat di hari ke-i didapat saat membeli coklat dengan harga terendah sebelum hari ke-i. Cara mudah untuk mencari harga terendah sebelum hari ke-i adalah dengan membuat for loop dari hari pertama sampai hari sebelum i.

Misalkan kita menjual pada hari ke-sellDay, berikut adalah potongan program untuk mencari keuntungan maksimum dengan menjual pada hari tersebut.

```
int maximumProfit = 0;
for (int sellDay = 1; sellDay < N; sellDay++) {
  int buyPrice = Integer.MAX_VALUE;
  for (int buDay = 0; buyDay < sellDay; buyDay++) {
    buyPrice = min(buyPrice, prices[buyDay]);
  }
  int currentProfit = prices[sellDay] - buyPrice;
  maximumProfit = max(maximumProfit, currentProfit);
}

return maximumProfit;
```

Kode tersebut memiliki kompleksitas sebesar $O(N^2)$. Total operasi yang dibutuhkan saat N = 500.000 adalah sekitar $2.5 \times 10^{11}$. Sementara grader pada umumnya, dalam 1 detik hanya dapat menjalankan sekitar $10^6$ sampai $10^8$ operasi[1]. Oleh karena itu diperlukan cara dengan kompleksitas yang lebih baik.

Perhatikan hal berikut. Misalkan kita telah mengetahui harga terendah pada rentang [0..sellDay], yaitu sama dengan minPrice. Apabila kita ingin mengetahui harga terendah pada rentang [0..sellDay + 1], kita tidak perlu melakukan iterasi dari awal. Harga terendah pada rentang [0..sellDay + 1] = min(minPrice, prices[sellDay + 1]).

[1] Jangan benar-benar dijadikan acuan. Bisa lebih maupun kurang.

Contoh:

| i | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| prices[i] | 4 | 3 | 5 | 2 | 3 | 10 | 7 | 3 | 2 | 3 | 4 |
| min_prices [0..i] | 4 | 3 | 3 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

Dapat dilihat min_prices [0..i] = min(min_prices [0..i-1], prices[i]).

Dengan adanya pengetahuan tersebut, bagian for loop yang di dalam, yang berfungsi untuk mencari harga minimum dari rentang 0 sampai sellDay - 1 dapat dihapus. Sehingga bentuk akhir dari program tersebut adalah seperti berikut.

```
int maximumProfit = 0;
int minimumBuyPrice = prices[0];
for (int sellDay = 1; sellDay < N; sellDay++) {
  int currentProfit = prices[sellDay] - minimumBuyPrice;
  maximumProfit = max(maximumProfit, currentProfit);
  minimumBuyPrice = min(minimumBuyPrice, prices[sellDay]);

  // perhatikan, update minimumBuyPrice harus setelah mencari keuntungan
  // pada hari tersebut, agar harga pada hari tersebut tidak menjadi
  // harga beli yang dipilih
}

return maximumProfit;
```

Kompleksitas program tersebut adalah O(N). Cukup untuk menyelesaikan input dengan N = 500.000 dalam waktu kurang dari 3 detik.