Perancangan dan Analisis Algoritma

DYNAMIC PROGRAMMING

HANUN SHAKA P -5025211051

Longest increasing subsequence

TESTING THE CATCHER

Link to problemset

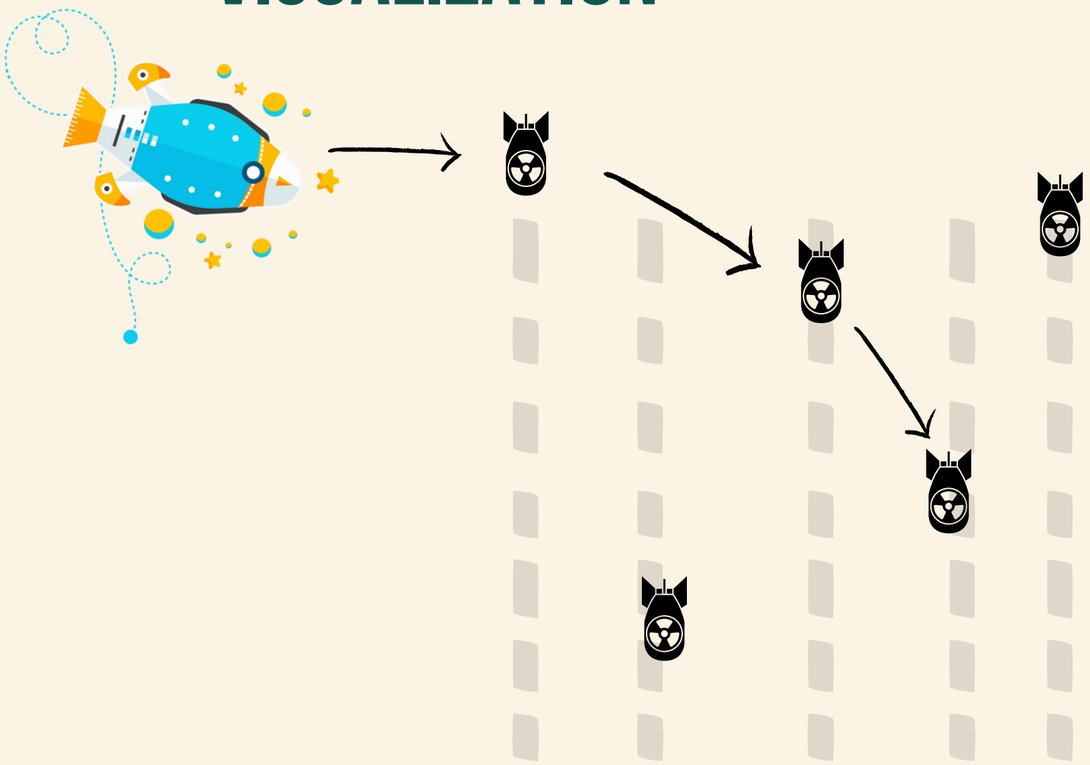
PROBLEM : TESTING THE CATCHER

Catcher merupakan misil defensif yang hanya dapat bergerak maju dan turun. Catcher dapat mengintervensi misil musuh dengan syarat :

1. Misil merupakan misil pertama yang akan diintervensi
2. Ketinggian misil tidak lebih dari ketinggian Catcher

Diminta menghitung sekuens misil terbanyak yang bisa diintervensi oleh Catcher.

VISUALIZATION



SAMPLE TEST CASE

389
207
155
300
299
170
158
65
-1
23
34
21
-1
-1

SAMPLE SOLUTION

Test #1:
maximum possible interceptions: 6
Test #2:
maximum possible interceptions: 2

Solution CODE SNIPPET

```
#include <iostream>
#include <stdio.h>
#include <vector>

int main() {
    int first_missile;
    int test_sequence = 1;

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...
    }
}
```

- LIBRARY DAN MAIN PROGRAM

Solution CODE SNIPPET

```
...

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        std::vector<int> missiles;
        missiles.push_back(first_missile);

        int missile;
        while(scanf("%d", &missile) != EOF && missile != -1) {
            missiles.push_back(missile);
        }
        ...
    }
}
```

- INPUT SEMUA MISIL MUSUH

LONGEST INCREASING SUBSEQUENCE

i = 1

j = 0

389	1
207	1
15	1
300	1
299	1
170	1
158	1
65	1

```
...

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...
        int n_missiles = missiles.size();
        std::vector<int> lis(n_missiles, 1);

        for(int i = 1; i < n_missiles; i++) {
            for(int j = 0; j < i; j++) {
                if(missiles[i] <= missiles[j] && lis[i] < lis[j] + 1) {
                    lis[i] = lis[j] + 1;
                }
            }
        }
      ...
    }
}
```

- **LIBRARY DAN MAIN PROGRAM**

LONGEST INCREASING SUBSEQUENCE

i = 1

j = 0

389	1
207	2
15	1
300	1
299	1
170	1
158	1
65	1

```
...

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...
        int n_missiles = missiles.size();
        std::vector<int> lis(n_missiles, 1);

        for(int i = 1; i < n_missiles; i++) {
            for(int j = 0; j < i; j++) {
                if(missiles[i] <= missiles[j] && lis[i] < lis[j] + 1) {
                    lis[i] = lis[j] + 1;
                }
            }
        }
      ...
    }
}
```

- **LIBRARY DAN MAIN PROGRAM**

LONGEST INCREASING SUBSEQUENCE

i = 2

j = 0

389	1
207	2
15	1
300	1
299	1
170	1
158	1
65	1

```
...

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...
        int n_missiles = missiles.size();
        std::vector<int> lis(n_missiles, 1);

        for(int i = 1; i < n_missiles; i++) {
            for(int j = 0; j < i; j++) {
                if(missiles[i] <= missiles[j] && lis[i] < lis[j] + 1) {
                    lis[i] = lis[j] + 1;
                }
            }
        }
      ...
    }
}
```

- INPUT MISIL

LONGEST INCREASING SUBSEQUENCE

i = 2

j = 0

389	1
207	2
15	2
300	1
299	1
170	1
158	1
65	1

```
...

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...
        int n_missiles = missiles.size();
        std::vector<int> lis(n_missiles, 1);

        for(int i = 1; i < n_missiles; i++) {
            for(int j = 0; j < i; j++) {
                if(missiles[i] <= missiles[j] && lis[i] < lis[j] + 1) {
                    lis[i] = lis[j] + 1;
                }
            }
        }
      ...
    }
}
```

- **LIBRARY DAN MAIN PROGRAM**

LONGEST INCREASING SUBSEQUENCE

i = 2

j = 1

389	1
207	2
15	3
300	1
299	1
170	1
158	1
65	1

```
...

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...
        int n_missiles = missiles.size();
        std::vector<int> lis(n_missiles, 1);

        for(int i = 1; i < n_missiles; i++) {
            for(int j = 0; j < i; j++) {
                if(missiles[i] <= missiles[j] && lis[i] < lis[j] + 1) {
                    lis[i] = lis[j] + 1;
                }
            }
        }
      ...
    }
}
```

- **LIBRARY DAN MAIN PROGRAM**

LONGEST INCREASING SUBSEQUENCE

389	1
207	2
15	3
300	2
299	3
170	4
158	5
65	6

```c
..

int main() {
    ...

    while(scanf("%d", &first_missile) != EOF && first_missile != -1) {
        ...

        int max = 0;
        for(int i = 0; i < n_missiles; i++) {
            if(lis[i] > max) {
                max = lis[i];
            }
        }

        printf("Test#%d: \n\tmaximum possible interceptions: %d\n", test_sequence, max);
        test_sequence++;
    }
}
```

- **LIBRARY DAN MAIN PROGRAM**

Knapsack

DIVIDING COINS

Link to problemset

PROBLEM : DIVIDING COINS

Ada 2 orang yang ingin membagi koin dalam sebuah tas seadil mungkin. Diminta mendapatkan selisih terkecil yang mungkin untuk pembagian koin antara kedua orang tersebut.

VISUALIZATION

SAMPLE TEST CASE

```
2
3
2 3 5
4
1 2 4 6
```

SAMPLE SOLUTION

```
0
1
```

Solution CODE SNIPPET

```
#include <iostream>
#include <algorithm>
#include <cstring>

using namespace std;

const int MAX_COINS = 100;
const int MAX_VALUE = 500;

int n;
int m[MAX_COINS + 1];
bool dp[MAX_COINS * MAX_VALUE / 2 + 1];

int main() {
    ...
}
```

- LIBRARY DAN MAIN PROGRAM

Solution CODE SNIPPET

```
...

int main() {
    cin >> n;
    while (n--) {
        int total = 0;
        memset(dp, 0, sizeof(dp));
        dp[0] = true;
        int numCoins;
        cin >> numCoins;
        ...
    }
    return 0;
}
```

- WHILE SEBANYAK TEST CASE

KNAPSACK

index	m
0	0
1	2
2	...
3	...

index	dp
0	true
1	false
2	false
3	false
4	false
5	false
6	false
7	false
8	false
9	false
10	false

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	...
3	...

i = 1

j = 2

dp[2] |= dp[2 - 2]

false |= true

dp[2] = true

index	dp
0	true
1	false
2	true
3	false
4	false
5	false
6	false
7	false
8	false
9	false
10	false

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	...
3	...

i = 1

j = 1

index	dp
0	true
1	false
2	true
3	false
4	false
5	false
6	false
7	false
8	false
9	false
10	false

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	3
3	...

i = 2

j = 2 + 3 = 5

dp[5] |= dp[5 - 3]

dp[5] |= dp[2]

dp[5] = true

index	dp
0	true
1	false
2	true
3	false
4	false
5	false
6	false
7	false
8	false
9	false
10	false

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	3
3	...

i = 2

j = 4

dp[4] |= dp[4 - 3]

dp[4] |= dp[1]

dp[4] = false

index	dp
0	true
1	false
2	true
3	false
4	false
5	true
6	false
7	false
8	false
9	false
10	false

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	3
3	...

i = 2

j = 3

dp[3] |= dp[3 - 3]

dp[3] |= dp[0]

dp[3] = true

index	dp
0	true
1	false
2	true
3	true
4	false
5	true
6	false
7	false
8	false
9	false
10	false

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	3
3	5

i = 3

j = 10

dp[3] |= dp[3 - 3]

dp[3] |= dp[0]

dp[3] = true

index	dp
0	true
1	false
2	true
3	true
4	false
5	true
6	false
7	true
8	true
9	false
10	true

```
...

int main() {
    ...
    while (n--) {
        ...
        for (int i = 1; i <= numCoins; i++) {
            cin >> m[i];
            total += m[i];
            for (int j = total; j >= m[i]; j--) {
                dp[j] |= dp[j - m[i]];
            }
        }
        ...
}
```

- **WHILE SEBANYAK TEST CASE**

KNAPSACK

index	m
0	0
1	2
2	3
3	5

total = 10

half = 5

diff = 10 - 2 * 5

diff = 0

index	dp
0	true
1	false
2	true
3	true
4	false
5	true
6	false
7	true
8	true
9	false
10	true

```
..

int main() {
    ...
    while (n--) {
        ...
        int half = total / 2;
        while (!dp[half]) {
            half--;
        }
        int diff = total - 2 * half;
        cout << diff << endl;
    }
    return 0;
}
```

- **WHILE HINGGA DIDAPAT TRUE**

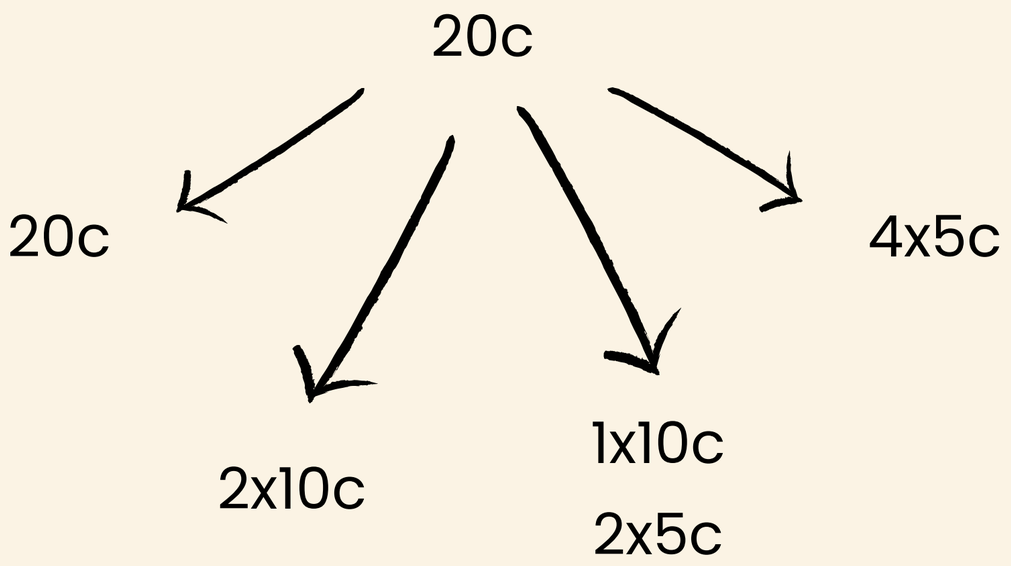
Coin Change
DOLLARS

Link to problemset

PROBLEM : DOLLARS

Pecahan uang Selandia Baru terdiri dari $100, $50, $20, $10, $5 and $2, $1, 50c, 20c, 10c and 5c. Nilai $1 sama dengan 100c. Diminta menghitung jumlah kombinasi susunan sebuah nominal uang

VISUALIZATION



SAMPLE TEST CASE

0.20
2.00
0.00

SAMPLE SOLUTION

0.20 4
2.00 293

Solution CODE SNIPPET

```
#include <iostream>
#include <cstdio>
#include <cstring>

using namespace std;

const int MAXN = 30005;

int main() {
    ...
}
```

- LIBRARY DAN MAIN PROGRAM

Solution CODE SNIPPET

```
...

int main() {
    int coins[] = {5, 10, 20, 50, 100, 200, 500, 1000, 2000, 5000, 10000};
    int n = sizeof(coins) / sizeof(int);
    long long dp[MAXN];
    memset(dp, 0, sizeof(dp));
    dp[0] = 1;
    ...
}
```

- INISIALISASI VARIABEL

KNAPSACK

index	dp
0	1
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0
10	0

index	coins
0	5
1	10
2	20
3	50
4	100
5	200
6	500
7	1000
8	2000
9	5000
10	10000

```
..

int main() {
    ...
    for (int i = 0; i < n; i++) {
        for (int j = coins[i]; j < MAXN; j++) {
            dp[j] += dp[j - coins[i]];
        }
    }
    ...
}
```

- **MENYUSUN ARRAY DP**

KNAPSACK

index	dp	index	coins
0	1	0	5
1	0	1	10
2	0	2	20
3	0	3	50
4	0	4	100
5	1	5	200
6	0	6	500
7	0	7	1000
8	0	8	2000
9	0	9	5000
10	1	10	10000

```
..

int main() {
    ...
    for (int i = 0; i < n; i++) {
        for (int j = coins[i]; j < MAXN; j++) {
            dp[j] += dp[j - coins[i]];
        }
    }
    ...
}
```

- **MENYUSUN ARRAY DP**

KNAPSACK

index	dp	index	coins
0	1	0	5
1	0	1	10
2	0	2	20
3	0	3	50
4	0	4	100
5	1	5	200
6	0	6	500
7	0	7	1000
8	0	8	2000
9	0	9	5000
10	2	10	10000

```
..

int main() {
    ...
    for (int i = 0; i < n; i++) {
        for (int j = coins[i]; j < MAXN; j++) {
            dp[j] += dp[j - coins[i]];
        }
    }
    ...
}
```

- **MENYUSUN ARRAY DP**

Solution CODE SNIPPET

```
...

int main() {
    ...
    double amount;
    while (scanf("%lf", &amount) == 1 && amount > 0) {
        int x = (int)(amount * 100 + 0.5);
        printf("%6.2f%17lld\n", amount, dp[x]);
    }
    return 0;
}
```

- INISIALISASI VARIABEL

Sekian

TERIMA KASIH

Link to presentation