Chocolate Bar DP Analysis - Manuel Morales

Analyze the problem based on sub-problems

This problem can be broken down into sub-problems by considering the weight of the knapsack. Each sub-problem represents the maximum number of pieces of chocolate that can be cut from a chocolate bar of weight W. The sub-problems can be solved by considering the weight of the chocolate bar and the weight of the pieces that can be cut from it.

Identify where the overlap occurs

The overlap occurse when we are calculating the maximum number of pieces of chocolate that can be cut from a chocolate bar of weight W. The maximum number of pieces that can be cut from a chocolate bar of weight W is the maximum of the maximum number of pieces that can be cut from a chocolate bar of weight W-1 and the maximum number of pieces that can be cut from a chocolate bar of weight W-2, W-3, ..., 1.

Implement the code

```
#ifdef LOCAL
#include "/home/morven/Desktop/code/competitive-programming/conf/debug.h"
#define line cerr << "-------------------" << endl;
#else
#define deb(x...)
#define line
#endif

#include <bits/stdc++.h>
using namespace std;

#define endl '\n'
#define F first
#define S second
#define pb push_back
#define sz size
#define all(x) begin(x), end(x)
#define sortt(x) sort(all(x))
#define each(x, xs) for (auto &x : (xs))
#define rep(i, be, en) for (__typeof(en) i = (be); i < (en); i++)
#define per(i, be) for (__typeof(be) i = (be)-1; i >= 0; i--)
#define readline(x) getline(cin, x)
#define strInt(str) stoi(str)
#define chrInt(chr) (int)chr - 48
#define ensp(i, n) (" \n"[i == n - 1])

template <typename... T>
void cinn(T &...args)
{
  ((cin >> args), ...);
}
```

```
template <typename... T>
void coutt(const T &...args)
{
  __typeof(sizeof...(T)) i = 1;
  ((cout << args << (i++ != sizeof...(T) ? " " : "")), ...);
  cout << '\n';
}

template <typename T>
void couts(const T &xs)
{
  for (__typeof(xs.sz()) i = 0; i < xs.sz(); i++)
  {
    cout << xs[i] << " \n"[i == xs.sz() - 1];
  }
}

using ll = long long;
using ld = long double;
using lli = long long int;
using pi = pair<int, int>;
using pl = pair<ll, ll>;
using ti = tuple<int, int, int>;
using tl = tuple<ll, ll, ll>;
using vi = vector<int>;
using vb = vector<bool>;
using vl = vector<ll>;
using vs = vector<string>;
using vvi = vector<vector<int>>;
using vvl = vector<vector<ll>>;
using vpi = vector<pair<int, int>>;
using vpl = vector<pair<ll, ll>>;
template <class T>
using pq = priority_queue<T>;
template <class T>
using pqg = priority_queue<T, vector<T>, greater<T>>;

const ll INF = INT64_MAX;
const int inf = INT32_MAX;
const ld PI = acos(-1);
const lli MOD = 1e9 + 7;
const vector<int> DX{1, 0, -1, 0}, DY{0, 1, 0, -1};
ll testId = 0;

void _()
{
  ios::sync_with_stdio(0);
  cin.tie(0);
  cout.tie(0);
#ifdef LOCAL
  freopen("/home/morven/Desktop/code/competitive-programming/conf/main.in",
"r", stdin);
  freopen("/home/morven/Desktop/code/competitive-
```

```
programming/conf/main.out", "w", stdout);
  freopen("/home/morven/Desktop/code/competitive-
programming/conf/main.err", "w", stderr);
#endif
}

void solve();

void init();

void exit();

int main()
{
  _();
  init();
  ll T = 1;
  // cinn(T);
  rep(t, 0, T)
  {
    testId++;
    solve();
  }
  return 0;
}

const ll MAXN = 202020;

void init()
{
}

void exit()
{
}

int knapsack(const int &W)
{
  vector<vector<int>> dp(W + 1, vector<int>(W + 1, 0));
  for (int i = 1; i <= W; i++)
  {
    for (int j = 1; j <= W; j++)
    {
      if (i <= j)
      {
        dp[i][j] = max(dp[i - 1][j], 1 + dp[i - 1][j - i]);
      }
      else
      {
        dp[i][j] = dp[i - 1][j];
      }
    }
  }
  return dp[W][W];
```

```
}

void solve()
{
  int W;
  cinn(W);
  coutt(knapsack(W));
}
```

What is the time complexity of your solution?

The time complexity of the solution is O(W^2) because we have a nested loop that iterates through the weight of the knapsack.