Part 1:
(a)
The path that has minimum travel time from node v to t under constraint B cost: it consists of node v and a path from node k_i to t. The path that has minimum travel time from node k_i to t: it consists of node k and a path from node p_i to t. And so on.
The optimal substructure for the problem is like finding the minimum time among paths that consist of node v and a path from k_i to t.

The value of the optimal solution: let OPT[v, b] be the minimum travel time from node v to t with cost at most b. let k_1, …, k_i be the next planet from v. let a_i be the travel time of the edge between k_i and v, b_i be the fuel cost of the edge between k_i and v.
OPT[v, b] = min{ OPT[k_1, b], a_1+OPT[k_1, b-b_1], …, a_i+OPT[k_i, b-b_i]}

(b) Algorithm (in psuedocode)
Input: G, s, t, a_e, b_e

```
n = number of nodes in G
for v = 1 to n
        for b = 0 to B
                if (v = t)
                        OPT[v, b] = 0, visited[v, b] = 1
                else
                        OPT[v, b] = ∞, visited[v,b] = 0
for b = 1 to B
        run dfs(OPT, visited, s, b)
return OPT[s, B]

def dfs(OPT, visited, s, b)
        if (visited[s, b] = 1)
                return
        else
                for k in s.nextplanet
                        dfs(OPT, visited, k, b - b_k)
                        OPT[s, b] = min{OPT[s, b], OPT[k, b - b_k]}
                Visited[s, b] = 1
```

(c)
Let V denotes the number of nodes, E be the number of edges.
Time complexity: $O((|V| + |E|) * B)$
Space complexity: $O(|V| * B)$

Part 2:
(a)
The optimal substructure:
Consider all subset of attacks, there can be two cases for every attack i:

1) the attack $f_i(t)$ is included in the optimal subset, $t\epsilon[0, T]$
2) the attack $f_i(t)$ is not included in the optimal subset, $t\epsilon[0, T]$

The value of the optimal solution: OPT(i, w) is the value of the maximum damage consisting of attacks from 1, 2, …, i with time limit w.

- Case1: the optimal solution does not select attack n
  - The optimal solution selects best of {1, 2, …, i-1} using time limit w.
- Case2: the optimal solution selects attack n
  - New time limit $w - w_i$
  - The optimal solution selects best of {1, 2, …, i-1} using the new time limit.

$$\text{OPT(i, w)} = \begin{cases} 0 & \textit{if } i=0 \\ OPT(i-1, w) & \textit{if } w_i>w \\ \max(OPT(i-1, w), OPT(i-1, w-w_i)+f_i) & \textit{otherwise} \end{cases}$$

(b) Algorithm (in psuedocode)
Input: n, f_0, f_1, …, f_{n-1}, $f_i(t)$, T

```
For i = 0 to T
        M[0, i] = 0

For i = 0 to n
        M[i, 0] = 0

For i = 1 to n
        For t = 0 to T
                M[i, t] = -∞
                For j = 0 to t
                        M[i, t] = max{ M[i, t], M[i − 1, t − j] + f_i(j) }

Return M[n, T]
```

(c) Algorithm for recreating the optimal solution

```
t = T
For ( i = n to 1 )
        For ( j = o to t )
                If M[i-1][j] + f_{i−1}(t) == M[i][t]
                        Print i, t-j
                        If (t=j) break
```

(d)
Time complexity: $\theta(nT^2)$
Space complexity: $\theta(nT)$