A picture containing text, font, screenshot, document

Description automatically generated

**Scratch**

Example milestones: 100, 200, 500, 950, 1800, 2250, 2500

**Problem Formulation**

T[i] = Minimum total penalty for reaching and stopping at T[i]

**Recurrence**

T[i] = Find one of the previous milestones such that the sum of minimum penalty to reach that milestone and penalty of reaching this milestone from that milestone is minimum.  
i.e.  
T[i] = minj0->i-1(T[j] + (200 - (ai-aj))^2)

K[i] = argminj0->i-1(T[j] + (200 - (ai-aj))^2), to track the previous milestone where we stopped for reaching i

**Base case**

T[0] = 0, corresponding to the case when you have not travelled any mile yet.

K[0] = 0

**Sample Run (rhymes with Temple Run)**

S = 100, 200, 500, 900, 1800, 2200, 2500   
T = 100, 0, 300^2, 300^2+400^2, 300^2+400^2+900^2, 300^2+400^2+900^2+400^2, 300^2+400^2+900^2+400^2+300^2

**Return Value**

The minimum penalty can be returned from last element in T i.e. T[n].

The optimal sequence of stops can be created by backtracking K array.

**PseudoCode**

T[0] = 0 ---------------------(1)  
S[0] = 0 ---------------------(2)

for i = 1->n ------------------------ (3)  
 min = inf  
 argmin = 0  
 for j = 0->n-1 --------------------------(4)  
 if (((S[i]-S[j])^2+T[j]) < min)  
 min = (S[i]-S[j])^2+T[j]  
 argmin = j

T[i] = min  
 K[i] = argmin

p = n

q = 1

While (p > 0) ------------------------------(5)  
 B[q] = S[p]  
 p = K[p]  
 q = q+1

Return reverse(B), T[n] --------------------------(6)

**Runtime Complexity**

Lines 1,2 take O(1) time  
Lines 3,4 take O(n) time each and since loop 4 is nested in loop 3, hence collectively they take O(n^2) time.  
Loop 5 takes O(n) in worst case if all milestone hotels are being used.  
Line 6 takes O(n) time due to reverse array function.  
  
Overall, it takes O(n^2) time.