## On The Run:

Switch between Highway A and Highway B at  $\mathbf{n}$  junctions. The time it takes to get from  $\mathbf{i}$  to  $\mathbf{i+1}$  junction =  $\mathbf{a(i)}$ , or  $\mathbf{b(i)}$ .  $\mathbf{k}$  = time to switch roads. Find fastest path.

- OPT:
  - OPT(j,PATH):
    - j = minimal time to junction j
    - PATH = Highway A or B the junction belongs to
    - cases:
      - 1. We are on PATH A
      - 2. We are on PATH B
- Identity:
  - OPT(j,PATH)=
    - if j=1
      - 0, because no previous paths
    - if PATH = A
      - MIN(a(j-1)+OPT(j-1,A), b(j-1)+k+OPT(j-1,B))
    - if PATH = B
      - MIN(b(j-1)+OPT(j-1,B), a(j-1)+k+OPT(j-1,A))
- Explanation:
  - Suppose we have found the shortest path to junction j-1 for PATH (either A or B)
  - If we are on path A,
    - we find the minimum between having traversed path A, or having traversed path B with an added cost of k
  - If we are on path B
    - we find the minimum between having traversed path B, or having traversed path A with an added cost of k
- Computation(j,PATH)=
  - o if j=1
    - return 0, because no previous paths
  - if M[j] is empty
    - if PATH = A
      - M[j]=MIN(a(j-1)+OPT(j-1,A), b(j-1)+k+OPT(j-1,B))
    - if PATH = B
      - M[j]=MIN(b(j-1)+OPT(j-1,B), a(j-1)+k+OPT(j-1,A))
  - return M[j]
- Solution(j):
  - M[n]=MIN(Computation(n,A), Computation(n,B));
  - o for i in (1 to n), print M[n].PATH.
- Run time:
  - Compute OPT(j): O(2n) for OPT(n,A) and OPT(n,B)
    - O(2n) = O(n)
  - Solution = O(n), linear loop

Overall = O(n)

## **Engine Trouble:**

**L** repairs are needed to repair your hog. Must be in specific order. Repair **i** will take **t(i)** hours. One person must do entire repair. Maggie charges **r** dollars/hour. Mikey takes flat rate **b** for 5 consecutive repairs. Find cheapest repair.

- OPT:
  - OPT(j, MECHANIC)=cheapest cost of repairs from 1 to j, with the current repairer MECHANIC, either Mikey or Maggie
    - case 1: Maggie is the current mechanic
    - case 2: Mikey is the current mechanic
- Identity:
  - OPT(j, MECHANIC)=
    - $\blacksquare$  if  $j \le 1$ ,
      - 0 since there were no previous repairs. Assuming we pay after the repair.
    - if MECHANIC = Maggie
      - r\*t(j) + MIN(OPT(j-1,Maggie), OPT(j-1,Mikey))
    - if MECHANIC = Mikey
      - b + MIN(OPT(j-5, Maggie), OPT(j-5, Mikey))
- Explanation:
  - Assume we pay Mikey when all his repairs are finished.
  - Suppose OPT(j-1, MECHANIC) returns the cheapest cost of all repairs from 1 to j-1, and MECHANIC is the mechanic that worked on that repair
  - If our current mechanic is Maggie, our current repair cost is the cost of Maggie (r\*t(j)) plus the previous repair j-1, which could have been from Maggie or Mikey
  - If our current mechanic is Mikey, our current repair cost is the cost of Mikey's constant (b) plus the previous repair path j-5, because b accounts for the last 5 repairs Mikey did, and previous to j-5, we could have either Maggie or Mikey do the repairs.
- Computation(j,MECHANIC):
  - $\circ$  if j <= 1,
    - 0 since there were no previous repairs. Assuming we pay after the repair.
  - if M[j] is empty
    - if MECHANIC = Maggie
      - M[j] = r\*t(j) + MIN(OPT(j-1,Maggie), OPT(j-1,Mikey))
    - if MECHANIC = Mikey
      - M[j] = b + MIN(OPT(j-5, Maggie), OPT(j-5, Mikey))
  - return M[j]
- Solution:
  - M[n]=MIN(Computation(n,A), Computation(n,B));
  - for i in (1 to n), print M[n].MECHANIC.

- Run time:
  - Compute OPT(j): O(2n) for OPT(n, Maggie) and OPT(n, Mikey)
    - O(2n) = O(n)
  - Solution = O(n), linear loop
  - Overall = O(n)

## **Achliopolis Vegan Hot Dog Eating Champion:**

Each day you will make **c(i)** from entering the contest. You must fast for 2 days before and after each contest. Find optimal schedule.

- OPT:
  - OPT(j)=optimal money made up to day j
    - case 1: We will enter the contest on this day
    - case 2: We will not enter
- Identity:
  - OTP(j)=
    - if j<=0, return
    - MAX(OPT(j-2)+c(j), OPT(j-1))
- Explanation:
  - Suppose OPT(j-1) is the most amount of money earned from the contest up to day j-1.
  - On day j, we will either enter the contest or not.
  - o If we don't, we will still have the money we earned to day j-1, so OTP(j-1)
  - If we do, the the nearest last day we could have entered the contest was j-2, so we add OPT(j-2)+c(j) to obtain the money earned up to day j.
- Computation(j):
  - if j<=0 return</li>
  - if M[j] is empty
    - M[j]=MAX(OPT(j-2)+c(j), OPT(j-1))
  - return M[j]
- Solution:
  - o for i in (1 to n)
  - if M[i] > M[i-1] (money was added)
    - We know we had entered the contest on the ith day.
    - print i
- Run time:
  - Compute OPT(j): O(n)
  - Solution = O(n), linear loop
  - Overall = O(n)