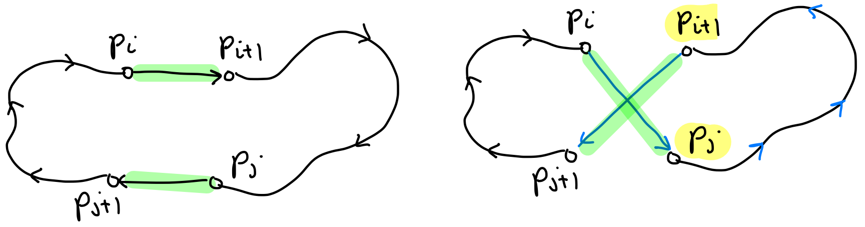
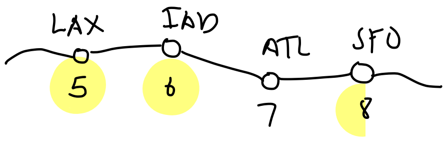
① reverse(i,j) → i<j : Reverse i+1 to j
j<i : Same as reverse(j,i)



② reverse("LAX", "SFO")

→ Use locator (AAXTree) to map labels to indices



(LAX, 5)
(IAD, 6)
(ATL, 7)
(SFO, 8)
→ AAXTree

→ Reversal requires you to replace old values with new

③ 2-Opt(i,j) – Apply reverse(i,j) only if cost strictly decreases

$\Delta(i,j) < 0$ where:

$$\Delta(i,j) = \left(\text{dist}^2(p_i, p_j) + \text{dist}^2(p_{i+1}, p_{j+1})\right) - \left(\text{dist}^2(p_i, p_{i+1}) + \text{dist}^2(p_j, p_{j+1})\right)$$

③' All indexing modulo n

④ 2-Opt-All

- Do 2-Opt(i,j) for all i,j

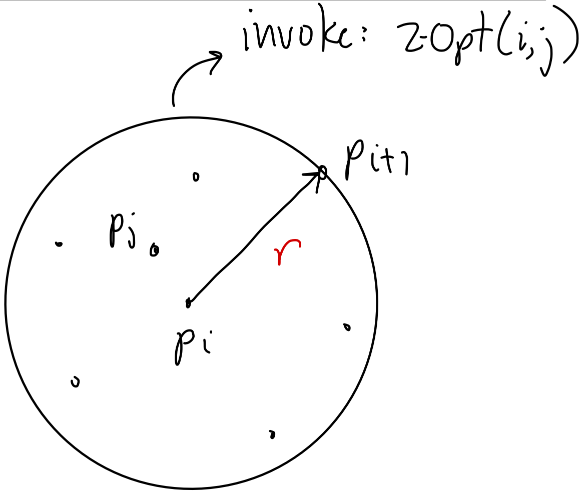
  $0 \leq i < j \leq n-1$

```
for (i = 0 to n-1)
    for (j = i+1 to n-1)
        2-Opt(i, j)
```

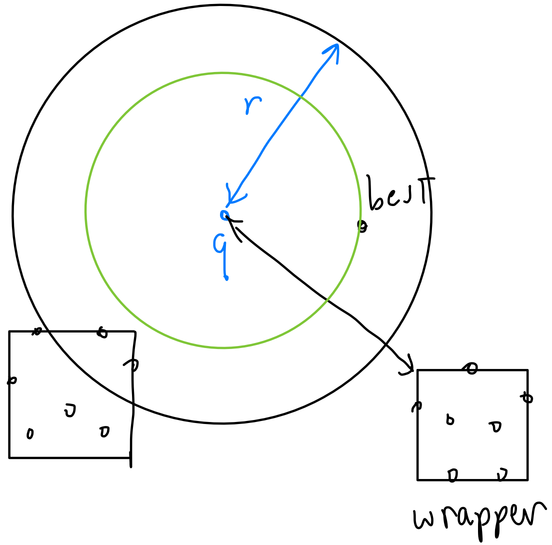
```
public twoOpt(String l1,
              String l2)
  → check validity
  → lookup l1, l2
      i1, i2 → exception
  → invoke helper
    twoOpt(int i1, int i2)
```

⑤ 2-Opt-NN (i)

- Do 2-Opt(i,j) where p_j is closest pt to p_i with dist(p_i, p_{i+1})

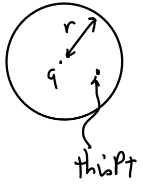


⑥ How to answer fixed-radius nearest neighbor query?



Helper:

```
LPoint fixedRadNN(Point2D q,
    double sqRad, LPoint best)
```

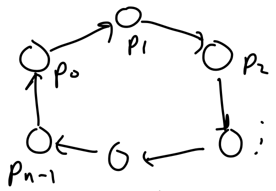


→ Extern: Check that pt is inside circle ($\text{dist}^2(\text{thisPt}, q) < r^2$) + better than best $\text{dist}^2(\text{thisPt}, q) < \text{dist}^2(\text{best}, q)$ ⇒ return thisPt (else return best)

→ Intern: If wrapper is outside
$\text{dist}^2(\text{wrapper}, q) \geq r^2$ → return best
If wrapper is not better than best
$\text{dist}^2(\text{wrapper}, q) > \text{dist}^2(\text{best}, q)$ → return best
else → Recurse:

```
best ← left.fixedRadNN(...)
best ← right.fixedRadNN(...)
```

⑦

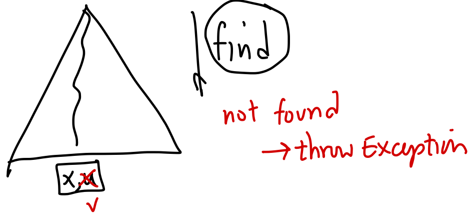


p[(i+1)%n]

double cost = $\sum_{i=0}^{n-1}$ p[i].distanceSq(p[i+1])

⑧ AAX Tree :

void replace(Key x, Value v)


find
not found
→ throw Exception
x, ~~x~~
v