Homework #1 Solution

Contact TAs: ada@csie.ntu.edu.tw

Problem 1

(1) Solve the recurrences.

(a) (3%) $f(n) = 16f\left(\frac{n}{4}\right) + 514n$
Base case: (1 points) Since they are the corner cases when $n < 4$, we need to prove all of them or choose a larger n_0.
We choose $n = 4$ for base case, for $c \geq \frac{530}{16}$

$$16f\left(\frac{n}{4}\right) + 514n = 16f(1) + 514n = 16 + 514 \leq 16c = c \times (4)^2 = cn^2$$

Induction: (2 points)
Assume that $f(k) \leq ck^2 - \frac{514}{3}k$ for $k < n$,

$$\begin{aligned} f(n) &= 16f\left(\frac{n}{4}\right) + 514n \\ &\leq 16\left(c\left(\frac{n}{4}\right)^2 - \frac{514}{3}\frac{n}{4}\right) + 514n \\ &= cn^2 - \frac{4 \times 514}{3}n + 514n \\ &= cn^2 - \frac{514}{3}n \end{aligned}$$

(b) (3%) $f(n) = 27f\left(\frac{n}{3}\right) + 40e^3n^3$
Tree: (1 points) Ignore the picture of drawing the tree. The cost of each layer is $40e^3n^3$ and the depth are $\log_3 n$. We can guess $n^3 \log n$.
Base case: $n = 3$, for $c \leq 867$

$$27f\left(\frac{n}{3}\right) + 40e^3n^3 = 27 + 40e^3 \leq c \leq cn^3 \log n$$

Induction:
Assume that $f(k) \leq ck^3 \log k - \frac{40e^3}{26}n^3$ for $k < n$,

$$\begin{aligned} f(n) &= 27f\left(\frac{n}{3}\right) + 40e^3n^3 \\ &\leq 27\left(c\left(\frac{n}{3}\right)^3 \log\frac{n}{3} - \frac{40e^3}{26}n^3\right) + 40e^3n^3 \\ &= cn^3 \log\frac{n}{3} - \frac{27 \times 40e^3}{26}n^3 + 40e^3n^3 \\ &= cn^3 \log n - cn^3 \log 3 - \frac{40e^3}{26}n^3 \\ &\leq cn^3 \log n - \frac{40e^3}{26}n^3 \end{aligned}$$

(2) Some proofs are not easily. You are not expected to proof them by your self, but you can estimate the order to get the answer.

Here are some useful equations. (Notice that, lg is 2 based logarithm and ln, log are e based logarithm)

$$n^{1/\lg n} \Rightarrow 2$$
$$f(n) = f(n-1) + \frac{1}{n} = \sum_{i=1}^{n} \frac{1}{i} \Rightarrow \text{ the sum of the reciprocals is } \Theta(\ln n)$$
$$e^{\ln n} = n$$
$$(\sqrt{2})^{\ln n} = n^{\ln \sqrt{2}} \approx n^{0.347}$$
$$\ln n! = \ln 1 + \ln 2 + \ldots + \ln n \Rightarrow O(n \lg n)$$
$$f(n) = ef(n/2) \Rightarrow e^{\lg n} = n^{\lg n} = n^{1/\log 2} \approx n^{1.442}$$
$$f(n) = f(n-1) + n^e \Rightarrow \sum_{i=1} i^e = O(n^{e+1}) \approx n^{3.718}$$
$$f(n) = f(n-1) + f(n-2) \Rightarrow f(n) = \frac{(\frac{1+\sqrt{5}}{2})^n - (\frac{1-\sqrt{5}}{2})^n}{\sqrt{5}} = O((\frac{1+\sqrt{5}}{2})^n) \approx 1.618^n$$
$$(\lg n)^{\ln n} = n^{\ln \lg n}$$
$$\{\ln \lg n = \ln 2 \times \lg \lg n\} < \{\lg \ln n = \lg(\lg n \times \ln 2) = \lg \lg n + \lg \ln 2\} < \lg \lg n$$
$$\text{where} \ln 2 \approx 0.693 \text{ and } \lg \ln 2 \approx -0.159$$

Then, we can get

$$\{n^{1/\lg n}, 2^{10000}, 2147483647\}$$
$$\{\lg \ln n\}$$
$$\{f(n) = f(n-1) + \frac{1}{n}, \sum_{i=1}^{n} \frac{1}{i}\}$$
$$\{(\sqrt{2})^{\ln n}\}$$
$$\{\frac{n}{\ln n}\}$$
$$\{\frac{10n}{e}, e^{\ln n}\}$$
$$\{f(n) = \sqrt{n}f(\sqrt{n}) + n, en \lg(\ln n)\}$$
$$\{n \lg n, n \ln n, \ln n!\}$$
$$\{f(n) = ef(n/2)\}$$
$$\{n^{3/2}\}$$
$$\{e^5 n^3 - 10n^2 + e^{1000}\}$$
$$\{f(n) = f(n-1) + n^e\}$$
$$\{(\lg n)^{\ln n}\}$$
$$\{n^{\lg \ln n}\}$$
$$\{n^{\lg \lg n}\}$$
$$\{f(n) = f(n-1) + f(n-2)\}$$
$$\{n!\}$$

Problem 2

(1) (a) If array A has a majority element, then after we split it in half, at least one of them has a majority element. We thus recursively divide A into halfs and if both of the sub-arrays have the same majority element x it returns x; if both are none (no majority element), it returns none; otherwise if the first half returns x and the second half returns y, we need to check whether the total number of x (or y) in A exceeds $n/2$. We have a divide and conquer scheme with a time complexity $O(n \log n)$.

(b) Pair up the elements of A to get $n/2$ pairs, then look at each pair: if the two elements are different, discard both of them; if they are the same, keep one of them; if the pair contains only one element x which is the only one left in this round, return x; otherwise, discard it. Repeat this procedure until no element is left and return none. Since we discard at least one element in each pair, there are at most half elements left after each round, so the time complexity will be $O(\frac{n}{2} + \frac{n}{4} + \cdots + 1) = O(n)$.

(2) (a) The merge algorithm has complexity $O(n + m)$, where n and m are the lengths of the two input sequences. If we merge the k arrays sequentially, the complexity will be $O(2n + 3n + 4n + \cdots + kn)$, which is $O(k^2 n)$.

(b) A better scheme is to merge in pairs in length n and then in length $2n$ and then $4n$, ..., until we hit kn. The merge complexity will be $O(2n(\frac{k}{2}) + 4n(\frac{k}{4}) + \cdots + kn(1)) = O(kn \log k)$.

Problem 3

Let N be the number of black points, B, W be the set of black/white points, $B_i, W_i \in \mathbb{R}^2$ be the position of the i-th black/white point.

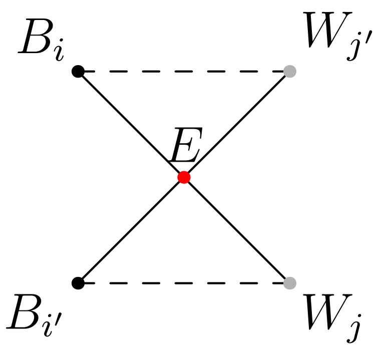
(1) (8%) For a specific match M, we can define its *total segment length* as

$$f(M) = \sum_{i=1}^{N} \text{len}\left(B_i W_{M(i)}\right)$$

where i-th black point is paired with $M(i)$-th white point.

Because the number of different matches is finite (exactly, $N!$), there must exist a match which *total segment length* is **minimum**. Call this match M_0.

If M_0 is a *miserable-word* match, some segments must intersect. Just say, there exists i, i', j, j' such that $M_0(i) = j, M_0(i') = j'$, and segment $B_i W_j$ intersects $B_{i'} W_{j'}$. More, since no three points lie on the same line, the two segment must intersects in their interior, and have exactly one common point. Let this common point be E.



By triangle inequality,

$$\text{len}(B_iW_j)+\text{len}(B_{i'}W_{j'}) = \text{len}(B_iE)+\text{len}(EW_{j'})+\text{len}(B_{i'}E)+\text{len}(EW_j) > \text{len}(B_iW_{j'})+\text{len}(B_{i'}W_j)$$

Swap these pairs to form a new match M_1, i.e. $M_1(i) = j', M_1(i') = j$, and $M_1(i) = M_0(i)$ otherwise. The new match will have shorter total length

$$f(M_1) = f(M_0) - \text{len}(B_iW_j) - \text{len}(B_{i'}W_{j'}) + \text{len}(B_iW_{j'}) + \text{len}(B_{i'}W_j) < f(M_0)$$

, caused a contradiction (M_0 is not minimum anymore).

So, M_0 must always be a *good-word* match.

In fact, there is an algorithm which finds M_0 in $O(N^3)$ time[1]. But that complexity is too *miserable* !!

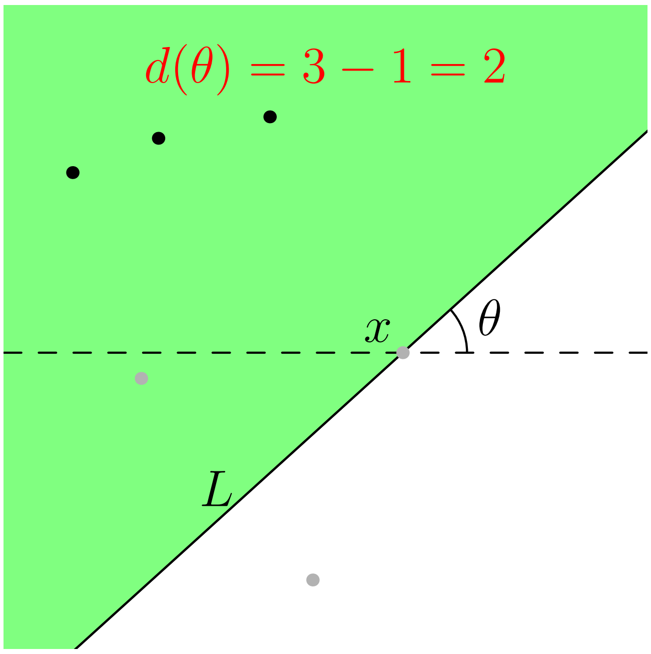
P.S. You can proof this fact using any method. Of course you can use the result of subtask (3) if you solved it, as construction is always a valid proof for existence!

(2) (15%) Choose an arbitrary white point as center point x.

[1]Hungarian algorithm, which can solve general **minimum weight maximum bipartite matching** problem in $O(N^3)$ time.

Let L be a line passing through x by angle θ (see figure next page). Among those points located on the "left" side of L (i.e. points which has **absolute angle**[2] between θ and $\theta+\pi$, with respect to x), define $d(\theta)$ be the difference of black and white points (not including x itself):

$$d(\theta) = (\# \text{ of black points}) - (\# \text{ of white points})$$



First, observe that $d(\theta+\pi)$ is the difference of the points on the "right" side of L (just rotate L by 180 degrees). There are N black and $N-1$ white points (other than x) in total, so $d(\theta) + d(\theta+\pi) = N - (N-1) = 1$.

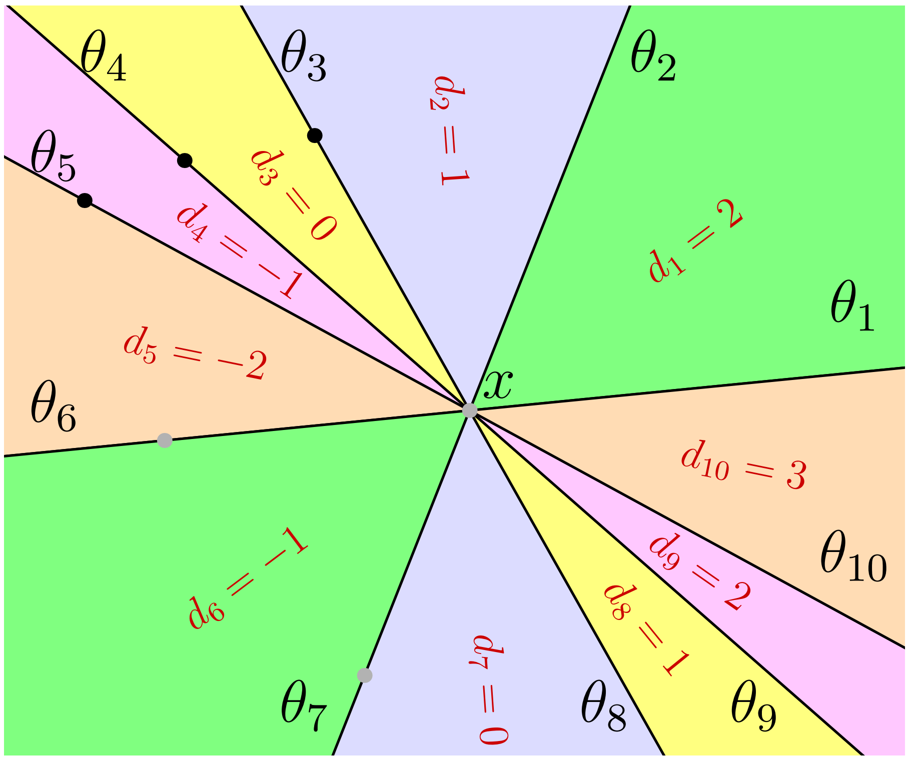
For a whole cycle ($\theta = 0$ to 2π), every point other than x will be passed by L for 2 times, and no two points will be pass simultaneously (since no 3 points lie on the same line), so there're $4N-2$ different angles $\theta = \theta_1, \theta_2, \cdots \theta_{4N-2}$ in ascending order, such that L passes through some point. These angles divide the whole cycle into $4N-2$ angle intervals, and $d(\theta)$ remains constant within each interval.

Let d_i be the value of $d(\theta)$ for interval $\theta_i < \theta < \theta_{i+1}$ [3]. By the above observation, we have $d_i + d_{i+2N-1} = 1$. More, at every angle θ_i, exactly one point enter or leave the left side of L, so $d_{i+1} = d_i \pm 1$.

Because $d_1 + d_{2N} = 1$, so $\min(d_1, d_{2N}) \le 0, \max(d_1, d_{2N}) \ge 1$, and d_i can only change by 1 each time, so there must exist $1 \le j \le 2N$ and $1 \le k \le 2N$ such that $d_j = 0, d_k = 1$ (like the discrete version of *intermediate value theorem*). More, because $N \ge 2$, we can choose j, k that at least one of them not on the boundary ($1 < j < 2N$ or $1 < k < 2N$), in other words, the *opposite* interval of $k : k + 2N - 1 \ne j$ has $d_{k+2N-1} = 0$. So we now have **two** different angle interval with $d = 0$.

It's obvious that the "left side" point set corresponding to these two interval can't be the same, so at least one of them have both side non-empty. Choose an arbitrary angle θ^* belongs to that interval, then the corresponding line L^* must be a valid line required by the problem.

[2]**Absolute angle** of point b with respect to point a is the angle rotated from $+x$ axis to ray $\overrightarrow{ab}$ counterclockwise.
[3]For simplicity, let $\theta_i = \theta_{i \bmod 4N-2}$ and $d_i = d_{i \bmod 4N-2}$.



Once existence proven, we can check for all $4N - 2$ angle intervals, and pick one with $d = 0$. The procedure follows:

1. List the splitting angles $\theta_1, \theta_2, \cdots, \theta_{4N-2}$, $O(N)$.
2. Sort them in ascending order, $O(N \log N)$.
3. Calculate d_1 by checking all the points, $O(N)$.
4. for $2 \le i \le 4N - 2$, calculate d_i by adding/removing one point from d_{i-1}. Each $O(1)$, total $O(N)$.
5. Choose some i that $d_i = 0$ and both side non-empty.
6. Choose an arbitrary angle within the interval (θ_i, θ_{i+1}), and output the corresponding liner.

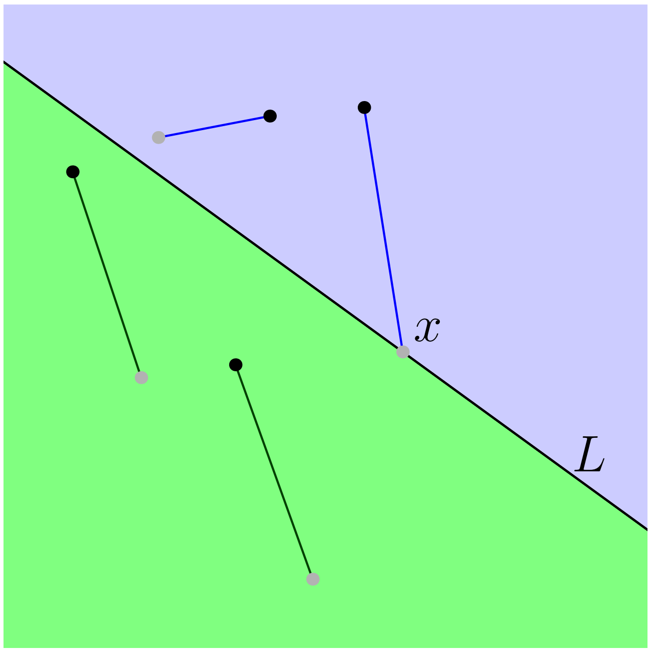
Total complexity: $O(N \log N)$.

(3) (7%)

- If $N = 1$, pair the only two points and we're done!
- If $N \ge 2$, choose a white point x and find a splitting line L in $O(N \log N)$ time, by method in (2). Divide S into two sets:
  - A : points on the left side of L (the side with $d = 0$)
  - B : point x, and the points on the right side of L

  By the property of L found in (2), both A and B are non-empty, and have number of black points equal to number of white points. That is, both A and B are valid sets as input sets (just like S) of the smaller subproblem ($N_A, N_B < N$).

  Get good-word matches for A and B by recursively solving the subproblems. All segments of pairs in A complete lie inside left side of L (not including L), and those of B complete lie inside right side of L (including L). So segments in A can't intersect with segments in B. Hence, combining their pairs directly results in a good-word match of S.



Correctness follows by induction. Total time complexity is $O(N^2 \log N)$, by solving the recurrence

$$T(N) = O(N \log N) + T(i) + T(N - i), 1 \leq i \leq N - 1$$

Problem 4 - Queue (Programming)

See the sample solution attached on the website.