Анализа проблема

Ако са x_i y_i означимо координате доњег левог темена правоугаоника са редним бројем i, а са xx_i і yy_i координате горњег десног темена правоугаоника са редним бројем i, онда ће кординате доњг левог темена обухватајућег правоугаоника након додавања правоугаоника број бити j

$$xo_j = \min\{x_{j-k+1}, x_{j-k+2}, ...x_j\}, \quad yo_j = \min\{y_{j-k+1}, y_{j-k+2}, ...y_j\}.$$

Јасно, координате горњег десног темена ће бити:

$$xxo_j = \max\{xx_{j-k+1}, xx_{j-k+2}, ...xx_j\}, \quad yyo_j = \max\{yy_{j-k+1}, yy_{j-k+2}, ...yy_j\}.$$

Ако је $j < k$, онда се формуле незнатно мењају те ће координате доњег левог темена бити

$$xo_j = \min\{x_1, x_2, ...x_j\}, \quad yo_j = \min\{y_1, y_2, ...y_j\}.$$

Аналогно би се одређивале координате горњег десног темена обухватајућег правоугаоника. Јасно, странице минималног обухватајућег правоугаоника, након -тог дана имају дужине

$$dx_j = xxo_j - xo_j, \quad dy_j = yyo_j - yo_j,$$

а дужине страница минималног обухватајућег правоугаоника за комплетан период од дана ће бити једнаке максимумима одговарајућих низова са дужинама.

Према томе, потребно је само одређивати минимуме (или максимуме) за сваких k узастопних елемената неког низа (низа састављеног од x или y координата темена правоугаоника). Описаћемо како одређујемо минимуме од сваких k узастопних елемената, а слично се одре|ују маскимуми.

Свакако се на први поглед намеће праволинијско решење у коме се за сваки подниз од k узастопних поново израчунава минимум и то решење има сложеност $\Theta(nk)$.

Мало профињење овог праволинијског решења добијамо тако што покушамо да искористимо претходно израчунати минимум. Наиме, претпоставимо да смо завршили обраду елемента x_i и већ израчунали минимум xo_i подниза од узастопних коме је последњи елемент x_i. Када се померимо на следећи елемент, треба да израчунамо xo_{i+1}. Ако је $x_{i+1} \leq xo_i$, онда ће xo_{i+1} бити једнако баш x_{i+1} ($xo_{i+1} = x_{i+1}$). Ако је пак $x_{i+1} > xo_i$, онда x_{i+1} не мора бити минимум од k последњих узастопних. Међутим, ако је $xo_i = x_{i-k+1}$, онда је претходни минимум био баш једнак елементу који "испада"иа блока узастопних и због тога треба поново израчунати минимум последњих k. Сложеност оваквог решења зависи од изгледа улазних података (тј. од тога колико често се дешава други случај), а у најнеповољнијем случају може бити $\Theta(nk)$.

Следећа варијанта решења се добија тако што се "актуелни елементи"(тј. последњих k обрађених) чувају у мин-хип-у. То обезбеђује да у константном времену одређујемо минимум од k узастопних. Међутим, треба имати у виду да при обради наредног елемента низа, тај елемент треба додати у хип и истовремено елемент који испада из блока актуелних избацити из хип-а. Сложеност ових операција је $\Theta(\log_2 k)$ (пошто се у хипу налази k елемената) па је тако сложеност комплетног решења $\Theta(n \log_2 k)$.

За најефкасније решење замислимо да смо цео низ поделили на дисјунктне блокове састављене од по k узастопних: први блок чине елементи од првог до k-ог, други од $k+1$-ог до $2k$-ог, трећи од $2k+1$-ог до $3k$-ог, итд. Тада ће сваки блок од тачно k узастопних бити састављен од делова из не више од два узастопна блока: десни крај једног блока (нека је то блок B_1) и леви крај наредног блока (нека је то блок B_2). Минимум тог блока се може лако израчунати ако су претходно израчунати минимуми за одговарајуће делове из блокова B_1 и B_2 (као мањи од та два минимума). Минимуми свих ових делова се могу једноставно израчунати са два пролаза кроз низ: један пролаз за рачунање минимума левих крајева и један пролаз за рачунање минимума десних крајева.

Прикажимо још један поступак за одређивање минимума за сваких k узастопних. Елемент x_i је кандидат за најмањи елемент поднизова (од k узастопних) који се завршавају на позицијама $i, i+1, ..., i+k-1$. Међутим, ако за x_j $(i < j)$ важи да је $x_j < x_i$, онда x_i престаје бити кандидат за минимум поднизова који се завршавају на позицијама $j, j+1, ..., i+k-1$ (значи може се елиминисати из разматрања). Можемо посматрати и обратно: након укључивања елемента x_j, сви елементи убачени пре x_j који имају особину да су већи (или једнаки) од x_j не могу бити више кандидати за минимум од k узастопних (без обзира што се налазе у блоку од k последњих узастопних). Због тога се могу избацити из разматрања сви елементе низа x који имају особину да су већи (или једнаки) од x_j. Приметимо да се тај елемент (x_j) додаје на крај листе кандидата (као последњи кандидат), зато што је он последњи додати и у једном тренутку сви пре њега додати ће бити избачени те он има шансе да буде најмањи у поднизу од последњих k након избацивања тих претходних. Елемент који се налази на почетку листе кандидата јесте баш најмањи за наредни блок узастопних. Али када у једном тренутку он буде на растојању бар k од елемента који се тренутно обрађује, он престаје да буде кандидат и зато се брише из те листе кандидата. Према томе, листа кандидата има особину да се са једне стране (почетка) елементи избацују када престану да буду актуелни, док се са друге стране и додају, али исто тако и избацују када се појави неки мањи.

Алгоритамске смернице

Ради илустрације могуће имплементације, прилажемо тело функције која као аргументе добија: дужину низова који садрже x координате левих крајева правоугаоника и x координате десних крајева тих правоугаоника (број n), број k, као и низове са x координатама левих крајева и x координатама десних крајева, а која враћа ширину минималног обухватајућег правоугаоника. Индексирање елемената низова креће од нула (0).

Једна могућа имплементација заснована на подели низа уз блокове дужине може имати следећи изглед:

```
int solve(int n, int k, int left[], int right[])
{
    int pref_min[MAXN], pref_max[MAXN];
    int suff_min[MAXN], suff_max[MAXN];
    for(int i = 0; i < n; i++)
    pref_min[i] = (i % k == 0) ? left[i] : min(pref_min[i - 1], left[i]);
    for(int i = n - 1; i >= 0; i--)
    suff_min[i] = (i % k == k - 1) ? left[i] : min(suff_min[i + 1], left[i]);
    for(int i = 0; i < n; i++)
    pref_max[i] = (i % k == 0) ? right[i] : max(pref_max[i - 1], right[i]);
    for(int i = n - 1; i >= 0; i--)
    suff_max[i] = (i % k == k - 1) ? right[i] : max(suff_max[i + 1], right[i]);
    int res = 0;
    for(int i = 0; i < n; i++)
    {
    int low = min(pref_min[i], suff_min[max(i - k + 1, 0)]);
    int high = max(pref_max[i], suff_max[max(i - k + 1, 0)]);
    res = max(res, high - low);
    }
    return res;
}
```

Имплементација заснована на формирању листе (реда) кандидата за минимум (максимум) од сваких k узастопних може имати следећи изглед:

```
int solve(int n, int k, int x1[], int x2[]) {
  int dxm, dxt;
  int qx1[MAXN], qx2[MAXN];
```

```
    int qx1s, qx1e, qx2s, qx2e;
    qx1s = qx2s = 0;
    qx1e = qx2e = 1;
    qx1[0] = qx2[0] = 0;
    dxm = x2[0] - x1[0];
    for (int i = 1; i < n; i++) {
        if (i - qx1[qx1s] >= k) qx1s++;
        if (i - qx2[qx2s] >= k) qx2s++;
        while ((qx1e > qx1s) && (x1[qx1[qx1e-1]] >= x1[i])) qx1e--;
        qx1[qx1e++] = i;
        while ((qx2e > qx2s) && (x2[qx2[qx2e-1]] <= x2[i])) qx2e--;
        qx2[qx2e++] = i;
        dxt = x2[qx2[qx2s]]-x1[qx1[qx1s]];
        if (dxt > dxm) dxm = dxt;
    }
    return dxm;
}
```