KSI 2013/2014

Úloha 3-1: Zapomenutý poklad

Jan Horáček
Gymnázium, Brno, Vídeňská 47; jan.horacek@seznam.cz

4. ledna 2014

1 Teoretický popis řešení

Mějme mapu o rozměrech $n * m$. Problém hledání pokladu v dvourozměrné mapě lze zjednodušit na problém hledání dvou bodů na dvou úsečkách reprezentující osy x a y. Pokud se budeme pohybovat v ose, druhá osa papouška vůbec neovlivní, takže lze bez problému nalézt jednu souřadnici. Pro druhou osu následně provedeme to samé.

Mějme neznámý bod $Y[x]$ reprezentující bod mapy na ose y o souřadnici x, který se pokusíme pomocí papouška nalézt.

K tomuto hledání zvolíme algoritmus, který aplikuje metodu půlení intervalů. Tento níže popsaný algoritmus má časovou složitost $O(logN)$, kde N je počet prvků na úsečce, tedy její délka. V našem případě je časová složitost $O(log(n))$.

V druhém kroku spustíme totožný algoritmus k nalezení y-ové souřadnice.

Celková časová složitost je tedy $O(log(n)+log(m)) = O(log(n*m)) = O(log(k))$, kde k je počet políček mapy.

2 Prohledávání metodou půlení intervalů

Mějme úsečku délky N a našeho milého papouška.

Na této úsečce se zeptáme papouška nejprve na bod ležící ve 2/3 její délky. Papoušek nám samozřejmě nic neřekne, protože se jedná o první požadavek na něj. Dále se ho zeptáme na bod ležící v 1/3 délky úsečky. Pokud nám řekne, že jsme blíže, je jasné, že je hledaný bod v levé půlce úsečky, pokud nám řekne opak, je jasné, že je hledaný bod v pravé půlce úsečky. Stejnou funkci pak zavoláme pro příslušnou polovinu úsečky. Ta nám vymezí oblast 1/4 z celkové délky úsečky, další iterace nám vymezí oblast 1/8 atd.

Metodu půlení intervalů končíme v momentě, kdy máme interval dlouhý 3 prvky, nebo méně. To z toho důvodu, že 3 prvky se špatně dělí na poloviny. Pokud bychom se o takové rozdělení pokusili, zaokrouhlení vnese do našeho algoritmu chybu, která se následně projeví špatným určením pozice pokladu, konkrétně se odchýlíme právě o jeden bod od správné pozice.

Pro 3 čísla pak položíme 2-3 dotazy na papouška, podle kterých přesně určíme absolutní pozici pokladu. Tento algoritmus může vypadat například takto: 2.

Funkce $papousek(intpos)$ vrací $true$, pokud jsme blíže, než v předchozím požadavku, jinak vrací $false$.

```
//pokud zbyvaji posledni tri, zavolame tuto fci
//tj. left, left+1, left+2
int tri(int left)
{
    papousek(left);
    if (!papousek(left+1)) return left;
    if (papousek(left+2)) return left+2; else return left+1;
}
```

Ve výše zmíněné rekurzivní funkci lze za účelem minimalizace požadavků na papouška s úspěchem implementovat jakési "pamatování si pozice posledního dotazu", abychom se na určitou pozici neptali několikrát zbytečně. Například pokud máme úsečku o délce N, zeptáme se na bod $2/3 * N$ a pak $1/3 * N$ a zjistíme, že poklad je vlevo, je poslední požadavek ve 2/3 levé poloviny, kterou dále analyzujeme. Funkce se tak nemusí ptát papouška na 2 body v levé polovině, protože ten vpravo je posledním dotazovaným bodem. Funkce se tak zeptá jen na bod vlevo, tedy v 16% celkové délky a z toho zjistí, jestli se poklad nachází v $0\ \% - 25\ \%$ úsečky, nebo v $25\ \% - 50\ \%$ úsečky.

3 Praktická realizace

V příloze *papousek.c* zasílám praktické řešení hledání bodu na úsečce metodou půlení intervalu v jazyce C.

4 Závěr

Velmi hezká úloha! Díky.