5. Správne poradie

Bruno Petrus - GBajkBa

Začneme tým, že jednotlivé závislosti medzi akciami uložíme do std::unordered_map. Map je key-value dátová štruktúra. Ako key sme dali vždy jednu akciu a do value sme uložili jednotlivé akcie, ktoré už nemôžme vykonať.
Napríklad:

```
4 3
4 6
```

4 bude key, 3 a 6 ako value. Takto analyzujeme zvyšné pravidlá. Vzhľadom na to, že jedna akcia môže mať viacej závislostí a std::map neumožňuje mať viac ako jednu value pre daný key, tak sme všetky values uložili do std::vector. Pre použitie std::unordered_map sme sa rozhodli, lebo nemusíme iterovať cez celú mapu, aby sme sa dostali k potrebným závislostiam (konštatná zložitosť).

Teraz môžeme začať riešiť program. Program iteruje cez jednotlivých bojovníkov. Vždy uloží ich akcie do std::unordered_map ako key a to či boli už vykonané (bool) do value. Pri každom bojovníkovi iteruje cez zoznam ich akcií dokiaľ má bojovník voľné akčné body. Pri rozhodovaní sa ktorú akciu má urobiť, sa pozrie na pravidlá. Ak by vykonaním danej akcie zablokoval vykonaniu ďalšej, nevykoná ju, ale pozrie sa na ďalšiu akciu. Ak existuje poradie v ktorom sa dajú vykonať všetky akcie, tak sa program nemôže zacykliť. Ak neexistuje tak sa zacyklí, lebo nebude vedieť, ktorú akciu má vykonať ako ďalšiu. Ak môže vykonať danú akciu, v mape nastaví value ako false, uloží do string, kde držíme riešenie a odpočíta zo zvyšných akčných bodov. Po nájdení riešenia pre všetkých bojovníkov vypíšeme riešenie.

Priestorová komplexnosť programu narastá lineárne so vstupom. Najväčší efekt na časovú komplexnosť má určovanie poradia akcií. Prejdeme cez všetkých *n* bojovníkov a pri každom musíme vykonať veľa porovnaní na základe toho, koľko má akčných bodov. Komplexnosť bude teda o niečo väčšia ako $O(n)$.