АНАЛИЗ НА РЕШЕНИЕТО НА ЗАДАЧА РАЗПИСАНИЕ

Задачата се свежда до реализиране на алгоритъм за съставяне на „разписание по нива“. Първо трябва да определим нивото на всяка операция, т.е. на всеки връх от дървото (това става с BFS). След това можем да се опитаме да действаме по следния начин: сортираме операциите в ненарастващ ред по това ниво и на всеки такт търсим в този сортиран списък *отляво надясно M* готови за изпълнение операции (ако няма толкова, вземаме колкото има). Увеличаваме времето с 1 и маркираме новите готови за изпълнение на следващи тактове операции, които са се получили след завършване изпълнението на операциите от предния такт. Този подход може да доведе до алгоритъм със сложност $O(N^2)$. Той е реализиран в **schedulen2.cpp.**

По-добрият алгоритъм е да се въведе отношение < между операциите, като операци1<операция2, ако нивото на операция 1 е < от нивото на операция 2. Използвайки това отношение се строи приоритетна опашка на готовите за изпълнение операции. Във върха на тази опашка винаги ще стои готова за изпълнение операция с най-голям номер на нивото. Първоначално в опашката се вкарват операциите от всички листа на дървото. На всеки такт първо се определя колко операции от опашката ще се изпълнят – това е min(*M*, *брой на операциите в опашката*). Тези операции се изваждат от опашката (все едно се изпълняват), като при всяко вадене на операция се увеличава с 1 броят на изпълнените предшественици на нейния баща в дървото. Ако, при това, операцията-баща става готова за изпълнение, то тя се запомня, за да бъде вкарана в приоритетната опашка след завършване на такта ***Важно:*** в никакъв случай готовата за изпълнение операция-баща не трябва да се вкарва веднага в приоритетната опашка, тъй като нейното ниво може да е такова, че да бъде извадена за изпълнение още в такта, в който е станала готова за изпълнение, което е невярно.

Тъй като всяка операция ще влезе и излезе в приоритетната опашка точно по веднаж и всяка операция по вкарване или изваждане на елемент от приоритетната опашка е със сложност O(*logk*), където *k* е броят на елементите в опашката, то общата сложност на алгоритъма е *NlogN.* Такъв алгоритъм е реализиран в **schedule.cpp.**

Автор: Руско Шиков