Řešení úlohy č. 5
Kuličkový počítač #3

Ve třetím kole bylo za úkol navrhnout dva podprogramy, z nichž jeden vyžadoval kromě implementace na kuličkovém počítači také algoritmický vhled. V minulém kole jsme navrhli užitečné operace pro manipulaci s binárním zápisem čísel. Zejména jsme si osvojili techniku, kdy postupně bitově posouváme číslo o jeden bit doprava a průběžně získáváme hodnotu posledního bitu binárního zápisu. Tak dokážeme z čísla extrahovat všechny bity jeho binárního zápisu. Tuto techniku budeme používat i pro návrh podprogramů v tomto kole.

Podprogram pro výpočet mediánu zadané rostoucí posloupnosti čísel

S podprogramy, které jsme navrhli v minulém kole, zvládneme tuto úlohu vyřešit poměrně snadno. Jako první musíme určit počet prvků zadané posloupnosti, tj. číslo n. To provedeme tak, že projdeme celý binární zápis zadaného čísla, které reprezentuje posloupnost, a spočítáme počet jedničkových bitů. Pro tento účel použijeme výše uvedený způsob získávání bitů z binárního zápisu daného čísla.

Jako další krok spočítáme index, na kterém se v posloupnosti nachází medián. Tento index je dle definice v zadání $\lceil \frac{n}{2} \rceil$, tj. jde o horní celou část při vydělení čísla n číslem 2. Pro výpočet horní celé části podprogram nemáme, ale poradíme si jednoduše – k n přičteme jedničku a provedeme celočíselné dělení tohoto čísla. To obecně můžeme udělat, jelikož jakékoliv sudé číslo x můžeme zapsat ve tvaru $2k, k \in \mathbb{Z}_0^+$, a platí tedy, že

$$\left\lfloor \frac{x}{2} \right\rfloor = \left\lfloor \frac{2k}{2} \right\rfloor = k = \left\lceil \frac{2k}{2} \right\rceil = \left\lceil \frac{x}{2} \right\rceil . \tag{1}$$

Dále víme, že každé liché číslo y můžeme zapsat ve tvaru $2l + 1, l \in \mathbb{Z}_0^+$. Platí tedy, že

$$\left\lfloor \frac{y}{2} \right\rfloor = \left\lfloor \frac{2l+1}{2} \right\rfloor = l = \left\lfloor \frac{2l}{2} \right\rfloor = \left\lfloor \frac{y-1}{2} \right\rfloor , \tag{2}$$

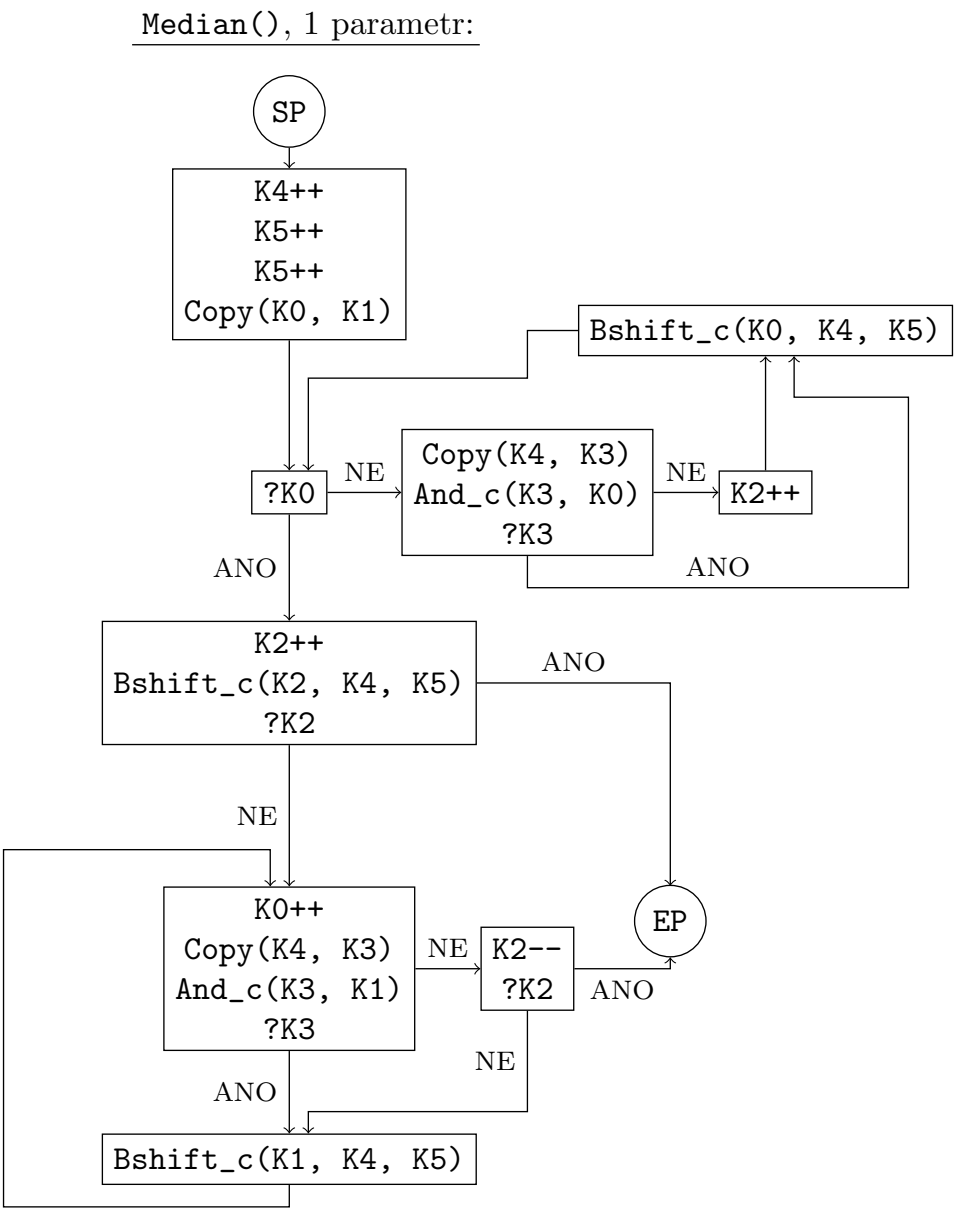
a dále také, že

$$\left\lfloor \frac{y+1}{2} \right\rfloor = \left\lfloor \frac{(2l+1)+1}{2} \right\rfloor = \left\lfloor \frac{2(l+1)}{2} \right\rfloor = l + 1 = \left\lceil \frac{2(l+1)}{2} \right\rceil = \left\lceil \frac{y}{2} \right\rceil . \tag{3}$$

Když tyto znalosti složíme dohromady, tak pro sudé n bude $n + 1$ liché a pro celočíselný podíl $n + 1$ číslem 2 bude platit, že $\lfloor \frac{n+1}{2} \rfloor =^{(2)} \lfloor \frac{n}{2} \rfloor =^{(1)} \lceil \frac{n}{2} \rceil$. Pro liché n bude přímo platit, že $\lfloor \frac{n+1}{2} \rfloor =^{(3)} \lceil \frac{n}{2} \rceil$. Přičtením jedničky k n a provedením celočíselného dělení číslem 2 tedy získáme $\lceil \frac{n}{2} \rceil$.

Poté nad původním číslem zopakujeme postup při zjišťování čísla n s tím, že si postupně budeme počítat i celkový počet zkontrolovaných bitů z. Jakmile poté dorazíme k $\lceil \frac{n}{2} \rceil$-nímu jedničkovému bitu, bude výsledkem právě číslo z. Speciálně ošetříme situaci kdy $n = 0$; tehdy vrátíme v kyblíčku K_0 nulu.

Kyblíčky K_0 a K_1 použijeme na práci se zadaným číslem a kyblíček K_0 navíc i na průběžné ukládání čísla z – pozdějšího výsledku. V kyblíčku K_2 budeme počítat číslo n, resp. $\lceil \frac{n}{2} \rceil$. Kyblíček K_3 použijeme pro zjišťování hodnoty posledního bitu čísla operací AND. V kyblíčcích K_4 a K_5 budeme držet konstanty, které se díky využití zálohovacích variant procedur za běhu podprogramu nezmění.



Komentář: Podprogram bere jako parametr jeden kyblíček K_0, reprezentující rostoucí posloupnost čísel. Jako výsledek vrátí podprogram v kyblíčku K_0 medián zadané posloupnosti.

Podprogram pro řešení úlohy přesného napětí

Jako první si tuto úlohu rozebereme z algoritmického hlediska. Kdybychom chtěli najít její řešení bez dalšího přemýšlení, museli bychom vyzkoušet všechny možné kombinace zdrojů s tím, že se nějaké zdroje mohou opakovat. Pro každou takovou kombinaci zdrojů bychom potom ověřili, zda dohromady v součtu dávají zdroje kýženou hodnotu napětí A. Toto by ovšem nevedlo na příliš efektivní algoritmus. Vždyť i kdybychom řešili úlohu jednodušší, kdy bychom každý zdroj mohli do součtu vybrat nejvýše jednou, tak bychom museli vyzkoušet všechny možné podmnožiny množiny zadaných zdrojů. I těch je ovšem exponenciálně mnoho – 2^n (kde n je počet zdrojů). V naší variantě úlohy se pak vyskytuje možných kombinací mnohem více, neboť každý zdroj může být navíc vybrán vícekrát. Ačkoliv v programech pro kuličkový počítač časovou složitost neřešíme, přeci jen by bylo příhodné sestavit algoritmus, který by byl efektivní a který bychom aplikovat i na větší instance problému.

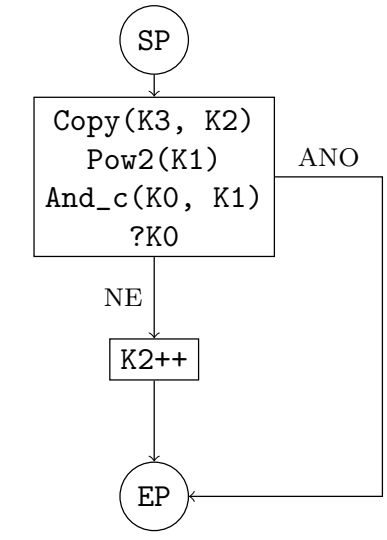
Pro řešení problému tedy použijeme techniku dynamického programování. Pro řešení uvažujme pole P(jak ho budeme reprezentovat v kuličkovém počítači si povíme později) o velikosti $A+1$, kde jeho políčka indexujeme od 0, tj. jejich indexy jsou z množiny $\{0, 1, 2, \ldots, A-1, A\}$. V každém políčku si budeme držet informaci, zda dokážeme nějakou kombinací zdrojů

získat napětí o velikosti rovné indexu daného políčka. V poli tedy bude v políčku na indexu i uložené číslo 1 právě tehdy, když jsme schopni pomocí zdrojů nakombinovat hodnotu napětí i. V opačném případě na daném políčku bude hodnota 0. Je zřejmé, že pokud se nám nějakým způsobem povede toto pole vyplnit korektně, bude se na konci stačit podívat na políčko na indexu A, tj. na $P[A]$, a zkontrolovat, zda je, či není nulové.

Jak tedy pole budeme vyplňovat? Uvažujme, že máme n zdrojů o napětích $c_1, c_2, \ldots, c_n$. Na začátku vyplníme celé pole nulami, až na políčko na indexu 0. Tam nastavíme $P[0] = 1$, jelikož nulové napětí jsme schopni získat triviálně tak, že nevybereme žádný zdroj. Nyní postupně pro všechny zdroje $1, 2, \ldots, n$ provedeme nad polem P následující postup. Pro j-tý zdroj o napětí c_j postupně v pořadí od nejmenšího indexu projdeme celé pole P. Pokud na políčku $P[i]$ nalezneme jedničku, nastavíme jedničku i v políčku $P[i + c_j]$. To protože pole P nám říká, že umíme nějak sestrojit napětí o velikosti i. Jelikož ale zrovna uvažujeme zdroj o napětí c_j a jelikož máme k dispozici nekonečně mnoho zdrojů o tomto napětí, tak víme, že jistě umíme sestrojit i napětí o hodnotě $i + c_j$. Navíc protože pole P procházíme v pořadí vzrůstajících indexů, zvládneme tímto postupem zachytit i situaci, kdy konkrétní zdroj použijeme vícekrát. To protože po nastavení políčka $P[i + c_j]$ na jedničku (pro nějaké i), nastavíme na jedničku i políčka $P[(i + c_j) + c_j]$, $P[((i + c_j) + c_j) + c_j], \ldots$, atd. Pro j-tý zdroj již navíc v poli P bude vyplněna informace pro všechny zdroje předcházející zdroji j. Pole P tak po zopakování tohoto postupu nade všemi zdroji zřejmě bude obsahovat korektní informaci o tom, jaká napětí v rozsahu 0 až A jsme ze zadaných zdrojů schopni sestavit. Speciálně se tak dozvíme, zda umíme sestavit napětí o velikosti A.

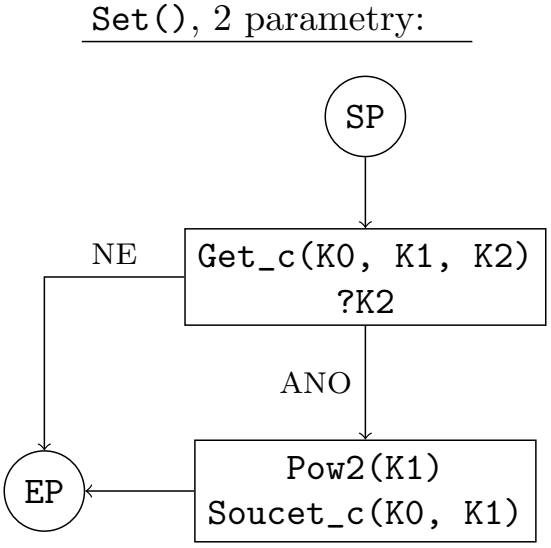
Nyní musíme rozmyslet, jak výše zmíněné pole P naimplementovat v rámci kuličkového počítače. Jednoduše však stačí pracovat s binárním zápisem nějakého čísla v kyblíčku jako s tímto polem. Poté bude i-té políčko pole P reprezentováno i-tým bitem daného čísla. Jelikož jsme v poli ukládali pouze jedničky a nuly, binární zápis čísla nám bude bohatě stačit. Budeme však potřebovat umět přistupovat k i-tému bitu nějakého čísla x. V řešení minulého kola jsme popsali jak toho dosáhnout – pro přístup k i-tému bitu vytvoříme si bitovou masku reprezentovanou číslem 2^i a provedeme operaci $x\&2^i$. Pokud je výsledek této operace nenulový, byl i-tý bit čísla x nastaven na jedničku a naopak. Pro tuto operaci si vytvoříme podprogram `Get()`, který pro daná dvě čísla a a b zjistí hodnotu b-tého bitu čísla a a v rámci zachování parametrů a a b nám výsledek uloží do jiného kyblíčku. Podprogram bude vypadat takto:

`Get()`, 3 parametry:



Komentář: Podprogram bere jako parametr tři kyblíčky. Kyblíčky K_0 a K_1 reprezentují dvě čísla a a b. Jako výsledek vrátí podprogram v kyblíčku K_2 jednu kuličku právě tehdy když b-tý bit čísla a je nenulový a nula kuliček v opačném případě.

V čísle, které reprezentuje pole P, taky budeme muset umět nastavit i-tý bit na jedničku. Opět jsme tento postup popsali v řešení minulého kola – pro nastavení i-tého bitu čísla x na jedničku stačí k tomuto číslu přičíst číslo 2^i. Toto však funguje pouze za předpokladu, že i-tý bit čísla x je původně nulový. Je-li však i-tý bit čísla x nastavený na jedničku, nemusíme vlastně číslo x nijak upravovat. Pro toto ověření můžeme použít již navržený podprogram `Get()`. Sestrojit program `Set()` na nastavení i-tého bitu nějakého čísla na jedničku tedy zvládneme jednoduše.

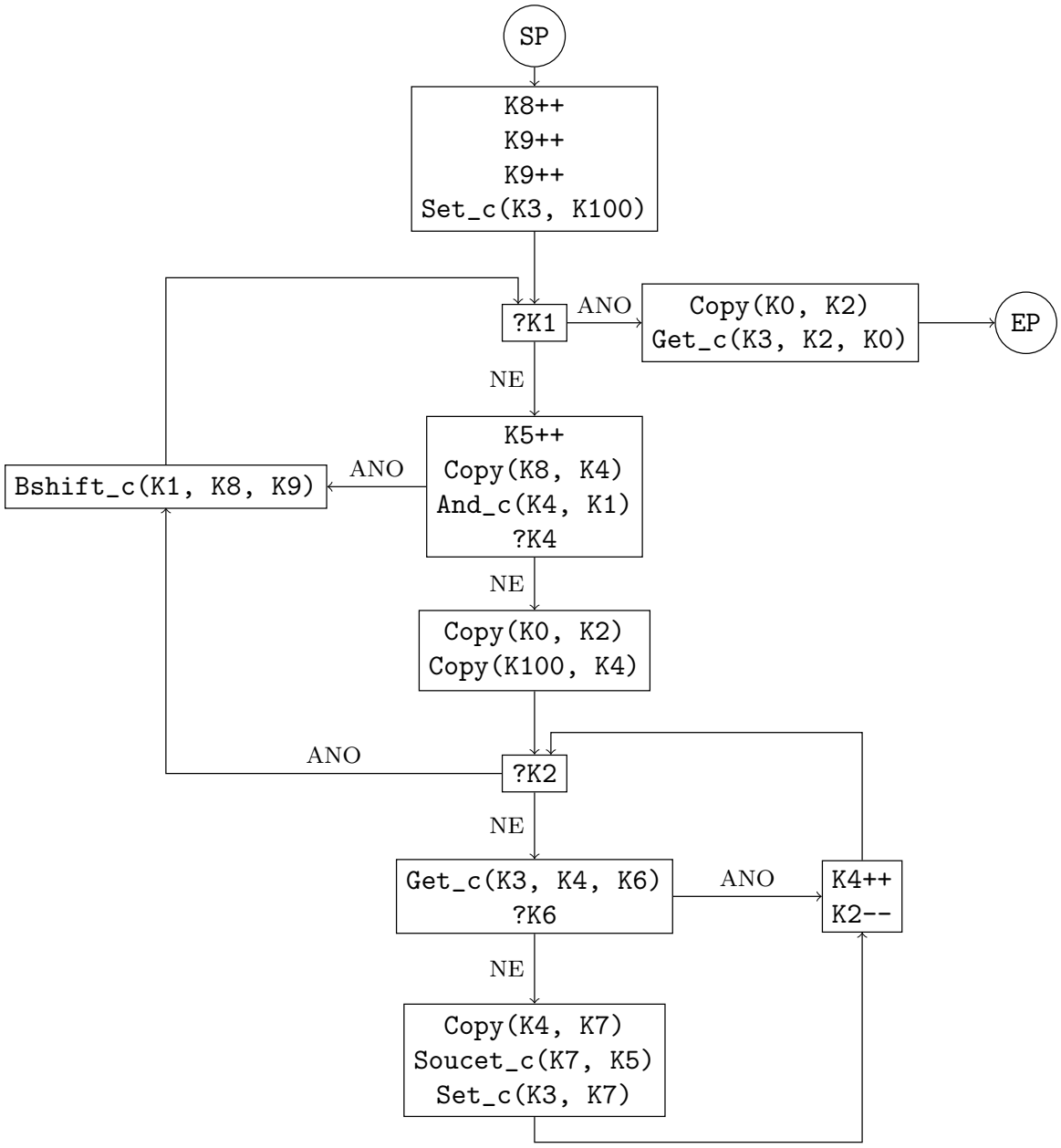


Komentář: Podprogram bere jako parametr dva kyblíčky. Oba kyblíčky K_0 i K_1 reprezentují dvě čísla a a b. Jako výsledek podprogram nastaví b-tý bit čísla a na jedničku.

Oba navržené podprogramy budeme používat ve variantách `Get_c()` a `Set_c()` pro zachování nenávratových parametrů.

Nic nám již nebrání v tom převést popsaný algoritmus do jazyka kuličkového počítače. Pole P budeme reprezentovat kyblíčkem K_3 a na začátku mu nastavíme na jedničku pouze 0-tý bit. To provedeme jednoduše voláním podprogramu `Set()` s nějakým určitě nulovým kyblíčkem – např K_100. Poté postupně v čísle c, které reprezentuje zdroje a které je uloženo v kyblíčku K_1, hledáme jedničkové bity a v kyblíčku K_5 si průběžně počítáme o kolikátý bit v čísle se jedná. Jakmile jedničkový bit nalezneme, jeho pořadí uložené v kyblíčku K_5 je hodnotou napětí tohoto zdroje a můžeme spustit smyčku pro vyplnění pole P, tj. bitů v kyblíčku K_3. To provedeme tak, že A-krát ověříme pomocí podprogramu `Get()` hodnotu i tého bitu čísla v kyblíčku K_3. Ačkoliv máme pole o velikosti $A + 1$, stačí nám ověřit A bitů pro $i \in \{0, 1, \ldots, A - 1\}$, jelikož z bitu na pozici A už expandovat další možnost řešení nemusíme – všechna větší napětí jsou pro naše řešení irelevantní. Nastavení správného indexu pro nastavení příp. další jedničky uděláme součtem čísla v kyblíčku K_4 (ten uchovává pořadové číslo bitu v kyblíčku reprezentujícím pole) a čísla v kyblíčku K_5 (hodnota napětí daného zdroje). Po zpracování všech zdrojů stačí nahlédnout podprogramem `Get()` na A-tý (číslováno od 0) bit čísla v kyblíčku K_3, což bude náš výsledek. V průběhu podprogramu si opět do kyblíčku K_7 a K_8 uložíme konstanty pro běh podprogramu.

Napeti(), 2 parametry:



Komentář: Podprogram bere jako parametr dva kyblíčky. Kyblíček K_0 reprezentuje číslo A a kyblíček K_1 reprezentuje rostoucí posloupnost čísel c. Jako výsledek podprogram vrátí v kyblíčku K_0 jednu kuličku právě tehdy, lze-li z čísel posloupnosti c složit přesně číslo A (při povolení nekonečně mnoho opakování konkrétního čísla). Podprogram tedy řeší úlohu přesného napětí tak, jak byla zadána.