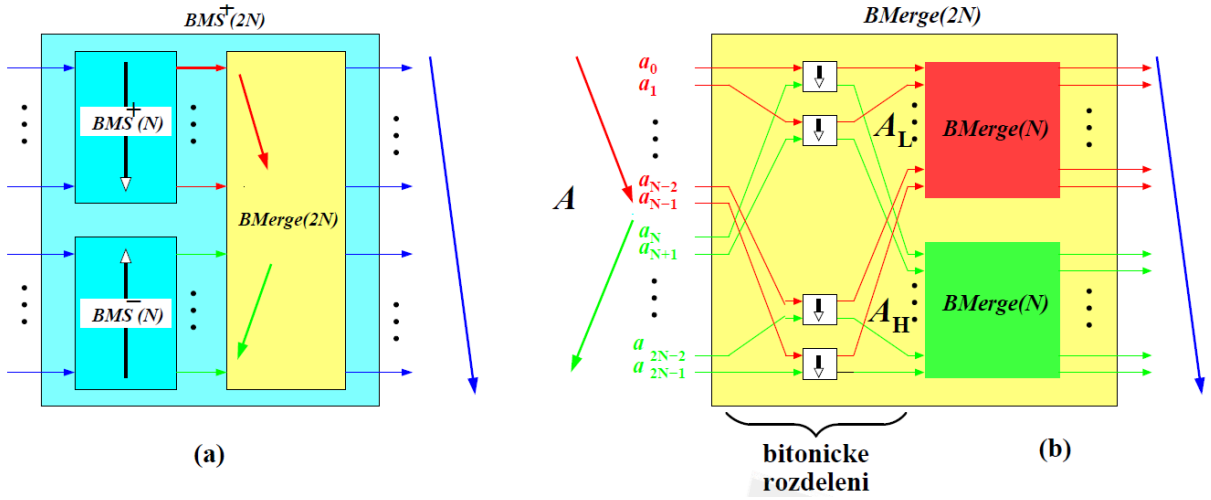
HW 3: Bitonický MergeSort - CUDA

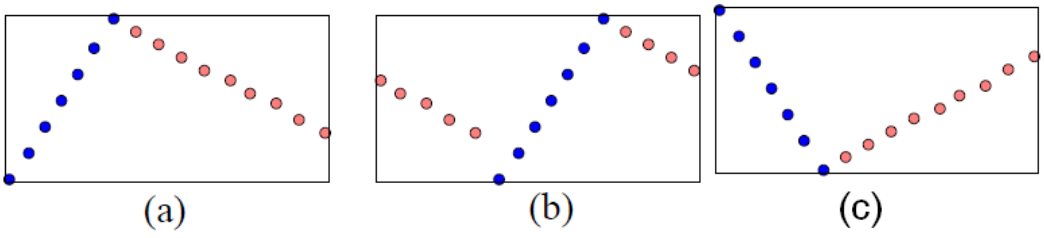
Bodů: 6 Termín odevzdání: do čtvrtka 15. 12. 2016 (12:00)

Naimplementujte v CUDA řadící algoritmus známý jako bitonický merge sort (BMS). Princip algoritmu je stejný jako u klasického merge sortu, až na fázi slučování seřazených posloupností, tj. dochází k rekurzivnímu dělení vstupního pole na poloviny, dokud neskončíme u polí velikosti 1 (viz obrázek 1a), tato pole se postupně po dvojicích (vzniklých dělením) slučují dohromady do větších, až nakonec dostaneme celé seřazené pole.



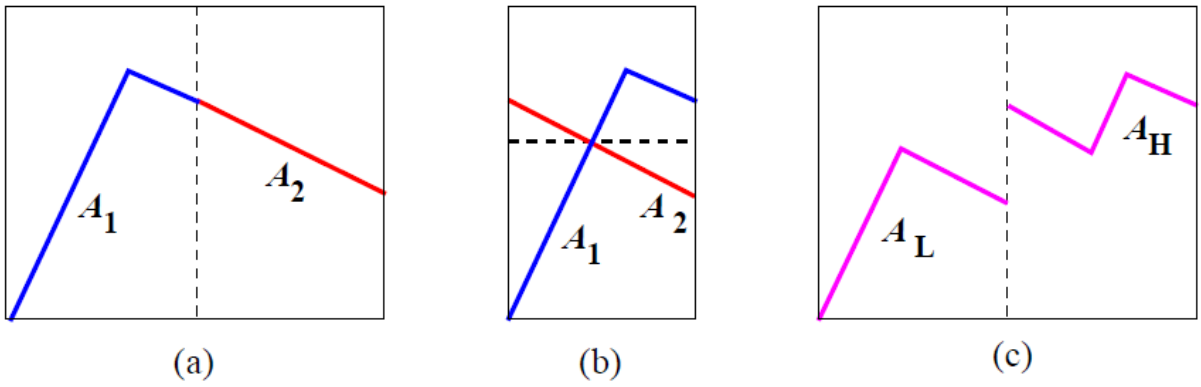
Obrázek 1: Bitonický merge sort a) fáze rozdělení BMS(2N) b) fáze slučování BMerge(2N).

U BMS je fáze slučování založena na operaci nazvané bitonické rozdělení. Tato operace předpokládá na vstupu bitonickou posloupnost (viz obrázek 2), tj. posloupnost skládající se ze dvou částí, kde jedna část monotónně roste (klesá), dosáhne svého maxima (minima) a poté monotónně klesá (roste) – např. 3 5 8 9 7 4 2 1, viz obrázek 2a) a 2c). Posloupnost je také bitonická, pokud ji lze cyklickou rotací na bitonickou upravit – např. 8 9 7 4 2 1 3 5 → 3 5 8 9 7 4 2 1, viz obrázek 2b).



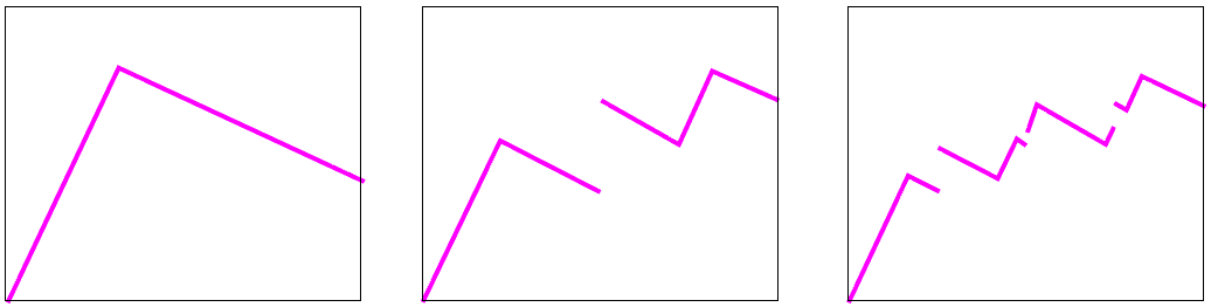
Obrázek 2: Příklady bitonických posloupností: a) monotónně rostoucí → maximum → monotónně klesající b) cyklickou rotací lze upravit na variantu a c) monotónně klesající → minimum → monotónně rostoucí.

Máme-li dánu bitonickou posloupnost $A = a_0, a_1, \ldots, a_{2N-1}$, pak její bitonické rozdělení je $A_0 = A_L A_H$, kde $A_L = \min(a_0, a_N), \min(a_1, a_{N+1}), \ldots, \min(a_{N-1}, a_{2N-1})$ a $A_H = \max(a_0, a_N), \max(a_1, a_{N+1}), \ldots, \max(a_{N-1}, a_{2N-1})$. Posloupnosti A_L a A_H vzniklé rozdělením jsou opět bitonické a každé číslo v A_L je menší než libovolné číslo v A_H (viz obrázek 3).



Obrázek 3: Příklad bitonického rozdělení bitonické posloupnosti A na dvě bitonické posloupnosti A_L a A_H.

Výsledkem bitonického rozdělení bitonické posloupnosti A jsou opět dvě bitonické posloupnosti A_L a A_H. Budeme-li dále rekurzivně aplikovat bitonické rozdělení na bitonické posloupnosti A_L a A_H, tak změníme posloupnost A na monotónní, tj. seřazenou (viz obrázek 4).



Obrázek 4: Rekurzivní aplikací bitonického rozdělení změníme posloupnost na seřazenou.

Rozvinutím rekurze BMS(2N) z obrázku 1a) získáme tzv. řadící síť, jejíž počet kroků je nezávislý na hodnotách řazených prvků a navíc se zde opakují nezávislé operace, díky čemuž je metoda vhodná pro paralelizaci (bloky s šipkami v BMerge(2N) označují komparátory, které prvky v případě potřeby prohodí tak, aby byly seřazeny – na jednom výstupu obdržíme min a na druhém max v závislosti na směru řazení – naznačen směrem šipky).

Pro implementaci BMS zvolte technologii CUDA a omezte se na posloupnosti, jejichž velikost je mocnina 2. Podrobnější popis BMS algoritmu a jeho sekvenční implementaci lze nalézt např. zde: http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/bitonic/bitonicen.htm.

Výsledný kód by měl obsahovat několik částí:

- Paralelní implementace BMS (CUDA).
- Sekvenční implementace řadícího algoritmu (nemusí být nutně BMS, klidně použijte quick sort, který je implementovaný v C).
- Porovnání výsledků sekvenční a paralelní implementace (tzn. ověření správnosti paralelní implementace).

Forma odevzdáni: emailem na adresu sloup@fel.cvut.cz se subjectem: "**A4M39GPU – HW3**".