[1] Odvoď Spodní mez pro OAB na $T(z_1,z_2,z_3)$, přepínání je WH, Startovní uzel uzly výstupně všeportový.

[One to All – ukolem bude rozeslat vsem tutez zpravu. Predpokladame, ze mame tolik vystupnich portu, jako sousedu (vseportovy). Pro 3D Toroid mame ***2n*** sousedu, kde *n* je pocet dimenzí tedy 3. Po prvnim kroku bude tedy informovano (2n + 1) uzlu tedy pocet sousedu + startovaci uzel startovaciho. (Jeste pripominam, ze vzhledem k uzlove symetrii Toroidu nezalezi na pozici startovaciho uzlu). V dalsim kroku posle kazdy z informovanych uzlu dalsim *2n* uzlum informaci. Informovano tedy v kroku ***k*** bude $\mathbf{(2n + 1)^k}$ uzlu. Celkem je v Toroidu $\mathbf{N = z_1 * z_2 * z_3}$ uzlu. Dostavame rovnici $\mathbf{(2n + 1)^k = N}$ – po jejim vyreseni pro ***n=3*** dostaneme $k = \log_7(z_1 \cdot z_2 \cdot z_3)$ **]**

[2] Odvodte vyraz pro prumer SE_n, $\Phi(SE_n)$ = ?

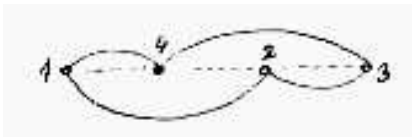
[Tento specialni graf zna pouze 2 moznosti, jak se posunout k sousedovi (resp. jak je definovana hrana)
a] rotovat bitovou posloupnost doleva ***rol****(x)*
b] negovat poslední (nejmene vyznamny) bit $\boldsymbol{neg_0(x)}$
Nejkratsi vzdalenost mezi nejvzdalenejsimi body grafu bude prumer. Evidentne nejvzdalenejsimi uzly jsou takove, které se lisi uplne ve všech ***n*** bitech. Abychom se od uzlu ***u*** dostali do ***v*** si v tomto pripade vyzada ***n*** operaci negace a (*n-1*) operaci rotace. Prumer SE_n tedy bude $n + n - 1 = (2n - 1)$]

[3] Dokažte, že v Q_4 mezi uzly u=1010 a v=0101 neexistuje Hamiltonovská cesta.

[Hamiltonovska cesta je takova permutace uzlu, ktera pro dany graf prochazi vsemi jeho uzly. V grafu s ***U*** uzly bude tedy evidentne mit ***U-1*** hran. Hyperkrychle ma $\boldsymbol{2^n}$ uzlu, libovolna Hamiltonova cesta tedy bude mit ($\boldsymbol{2^n-1}$) hran, coz je liche cislo. Nyni uvazujme: lisi-li se 2 uzly od sebe v danem bitu, musi cesta mezi nimi v odpovidajici dimenzi mit lichy pocet hran. Pokud se v danem bitu nelisi, pak v dane dimenzi bude libovolna cesta, ktera tyto body spojuje mit sudy pocet hran. Nase cesta tedy bude mit lichy pocet hran v 1., 2., 3., i ctvrte dimenzi, protoze se ve vsech techto dimenzich ***u*** a ***v*** vzajemne lisi. Jina dimenze v krychli Q_4 neni, v opacnem pripade by pocet hran v techto dimenzich byl sudy. Cesta ktera bude mezi nasemi body ***u*** a ***v*** bude mit ve vsech dimenzich lichy pocet posuvu. Celkem to tedy bude $4 \cdot liche\_cislo = sude\_cislo$.
Jenze aby dana cesta mezi temito uzly byla Hamiltonovska, musela by mit lichy pocet hran (musi projit vsemi uzly bez opakovani) – viz. vyse. Mezi nasimi uzly tedy neexistuje cesta liche delky a proto neexistuje ani Hamiltonovska cesta mezi temito uzly. **]**

[4] Dokažte, že 2D Toroid $T(z_1, z_2)$ je výpočetně ekvivalentní s $M(z_1, z_2)$

[2 site G a H jsou kvaziizometricke ⇔ lze jednu do druhe vnorit s konstantnimi parametry vnoreni.
Jsou vypocetne ekvivalentni ⇔ lze jednu simulovat na druhe s nejvyse konstantnim zpozdenim.
Z kvaziizometrie tedy evidentne plyne vypocetni ekvivalence.
a] Vnoreni 2D Mrizky do 2D Toroidu je trivialni (Toroid je vytvoren z mrizky stejneho rozmeru pridanim nekolika hran)
b] Vnoreni 2D Toroidu do 2D Mrizky je taky velice jednoduche, nicmene vyzaduje nasledujici uvahu [viz. obrazek] – Dosahneme tak vnoreni s konstantnimi koeficienty vnoreni, coz ve vysledku implikuje vypocetni ekvivalenci. (Load=1, Dil=Encg=2 - vyplyva z obrazku)**]**



[5] Urcete spodni mez pro pocet kroku operace AAB na vseportovem 3D Toroidu.

[Toroid je uzlove symetricky, nezajima nas tedy, kde je zacatek (který je inicializacni uzel).
All to All Broadcast, tedy ke zdarnemu cili musí každý uzel prijmout od kazdeho kolegy zpravu. Uzlu v danem 3D Toroidu bude: $U = z_1 * z_2 * z_3$, kde z_i je velikost i-te dimenze Toroidu. V jednom kroku nemuze uzel prijmout vice zprav, nez ma sousedu, tedy 2n, kde n je počet dimenzi grafu, v nasem pripade 3. Aby ziskal celkem U-1 zprav, potrebuje k tomu (U-1)/2n kroku.

[6] Proc nelze CBT_n pro n>1 vnorit do Q_{n+1} hyperkrychle s load=dil=1 ?

[Jestlize ma existovat vnoreni s temito parametry, musel by byt vnorovany graf podgrafem grafu, do ktereho se snazi vnorit (musi byt CBT(n) podgrafem Q(n+1)), coz neni. Zatimco hyperkrychle je klasicky pripad bipartitniho grafu obarvitelny dvema barvami (polovina uzlu bude cerna a polovina bila), uplny binarni strom neboli CBT bipartitni neni. Z totoho jiz primo vyplyva, ze nemuzeme vnorit CBT do Q ani v pripade, ze je Q o jednu dimenzi vetsi a ma tedy vice uzlu. Nebudeme-li trvat na dil=1, pouziva se s uspechem technika "zdvojeni korene" – [viz priklad 31]**]**

[7] Charakterizujte uzlovou a hranovou symetrii 3D Toroidu $T[z_1, z_2, z_3]$

[Uzlova / Hranova symetrie si vyzaduje jako podminku sve existence uzlovy / hranovy automorfismus. Zatimco v Toroidu lehce nalezneme uzlovy automorfismus *přeložení* , definovane jako $\tau(x) = x \oplus w$. Podle jednoduche binarni aritmetiky $\tau(u) = v; \tau(v) = u$. 3D Toroid je tedy uzlove symetricky. Pokud $z_1=z_2=z_3$, pak take rotace bude automorfismem a tedy takovyto specialni pripad Toroidu by take byl hranove symetricky.**]**

[8] Automorfismy Q_n. Kolik jich je a proc?

[V hyperkrychli mame 2 moznosti jak dosahnout automorfismu. Jednak je to *rotace* predstavitelna jako libovolna permutace(prejmenovani) dimenzi a jednak je to *přeložení*. Prvni moznost nam dava celkem ***n!*** moznosti jak vytvorit automorfni zobrazeni. Prelozeni je definovano jako $\tau(x) = x \oplus u \oplus v$. Body ***u*** a ***v*** jsou body z hyperkrychle, proto jsou tedy n-bitove. Jejich XOR $\oplus$ nam vrati opet n-bitovou hodnotu. Vidime tedy, ze bude presne 2^n moznosti, jak udelat v hyperkrychli prelozeni. Nyni ziskame automorfismus libovolnou kombinaci techto dvou metod (rotace a prelozeni) mame tedy dohromady ($\mathbf{n \times 2^n}$) automorfismu.]

[9] Vykonove porovnej PRIORITY/COMMON resp PRIORITY/ARBITARY CRCW.

[PCRCW >= CCRCW

Vsechny CRCW se pri cteni chovaji stejne. Pri zapisu Prioritni vyuziva priorit jednotlivych procesoru a pouze nejprioritnejsi ma povoleno do sdilene pameti zapisovat. CCRCW predpoklada, ze vsechny procesory chteji zapsat jednu a tutez hodnotu, v opacnem pripade nastava chyba (v pameti nedef. hodnota). Z toho tedy plyne, ze CCRCW lze pomoci PCRCW simulovat naprosto bez zpomaleni. Naopak, chteli-li bychom simulovat PCRCW pomoci CCRCW, museli bychom algoritmem zavest pomocne priority, umele vyhodnotit nejvyssi z tech, co chteji zapisovat a umele zakazat zapis tem, co maji prioritu nizsi, coz by zpusobilo zpomaleni.

PCRCW >= ACRCW

Pri vice zadostech o zapis je nahodne vybran jeden procesor. Nijak neni blize urceno, ktery to ma byt. Muze to tedy byt klidne ten s nejvetsi prioritou, coz je priklad PCRCW. Prioritni muze tedy nahodny simulovat beze ztraty vykonu. Aby dokazal ACRCW vyuzivat priority, bylo by nutne napr. explicitne pozdrzovat ostatni procesory pri zapisu, coz by melo za nasledek vysledne zpomaleni...

[10] Jaka je spodni mez pro pocet kroku pro Q_n a AAB ?

[Potrebujeme prijmout ***U-1*** zprav od U-1 kolegu. V hyperkrychli je celkem $U=2^n$ uzlu, z nichž každý ma ***n*** sousedu. Z cehoz jiz plyne, ze bude potreba $\frac{2^2-1}{n}$ kroku.

[11] Urcete a dokazte chybovy prumer hyperkrychle Q_n.

[chybova vzdalenost uzlu u a v je maximum z delek vsech nejkratsich uzlove disjunktnich cest. Chybovy prumer je maximalni chybova vzdalenost v grafu. z definice je to maximum ze vzdalenosti uzlove disjunktnich cest.

Mezi dvema uzly ve vzdalenosti k je v krychli ***k cest delky k*** a ***n-k cest delky n-k+2***. Kdyz si vezmes dva uzly lisici se v ***n*** bitech, tak dostanes jasne, ze chybova vzdalenost je $\boxed{n}$]

[12] Kolik permutaci udela neprima sit stupne k s n vstupy s pouzitim prepinace 2x2?

[mam $\frac{n}{2}$ dvojic. Kazda ma 2 moznosti, takze $2^{\frac{n}{2}}$ a ted mam ***k*** urovni, takze ***k*** moznosti v kazde urovni jak tech $2^{\frac{n}{2}}$ dal usporadat. Vysledek tedy $\boxed{\left(2^{\frac{n}{2}}\right)^k = 2^{\frac{kn}{2}}}$]

[13] Odvodte prumer site wBF_n

[Jedna se o *zabaleného motýlka*. Uzel je dvojice (i, x) : 0 <= i <= n, x je n-bitovy vektor

Hrana je v nem jenom tam, kde:

a] x=y & j = i ⊕ 1

b] j=i ⊕ 1 & y=neg_i x (i, x) = (0, 010), (j, y)= (1, 011)

Pokud se dva uzly lisi v ***n*** bitech, je potreba ***n*** "*hyperkubickych hran*" ke zmene x => y. Provede se tedy ***n*** negaci a ***n*** krat se take provede ⊕ 1 na hodnotu ***i***. XOR ve vysledku vyhodi stejnou hodnotu, jako tam byla na zacatku, takze se dotaneme z (i, x) do (i, y) po ***n*** krocich. V nejhorsim pripade jeste nasleduje $\lfloor n/2 \rfloor$ kroku v kruznici => celkem max= n + $\lfloor n/2 \rfloor$. Prumer xBF_n = $\boxed{n + \left\lfloor \frac{n}{2} \right\rfloor}$**]**

[14] Definujte NP/P/NC tridu.

[NP – Mnozina vsech jazyku verifikovatelnych v polynomickem case => Pro kazde slovo z tohoto jazyka existuje *certifikát* takovy, ze algoritmus A – A(slovo, certifikat) = 1 => A verifikuje dany jazyk L_A

P – je mnozina vsech jazyku rozhodnutelnych v polynomickem case P ⊂ NP
jazyk L_1 je rekukovatelny na L_2 v polynomickem case <=> existuje funkce r: $\{0,1\}^* \rightarrow \{0,1\}^*$, ze kazde slovo x jazyka L_1 <=> r(x) ∈ L_2 a funkce r lze spocitat v polynomickem case.

L_1 je **P-redukovatelny** na L_2 <=> A_1 lze popsat jako polynomialne slozity algoritmus, ktery vola A_2

NPC – mnozina NP uplnych jazyku, tedy jazyku P-redukovatelnych

NC – mnozina jazyku rozhodnutelnych maximalne v polylogaritmickem case na PRAM s nejvyse polynomialnim #procesoru (binarni redukce, prefix soucet, konstrukce Eul. cesty)

Jazyk L_1 ∈ P je **NC redukovatelny** na L_2 <=> existuje funkce r $\{0,1\}^* \rightarrow \{0,1\}^*$ a pro kazde slovo x (x∈ L_1 <=> r(x) ∈ L_2) & r je nejvyse NC

Jazyk L_1 je **P-uplny** <=> libovolny jazyk L'∈P na nej lze NC-redukovat (hodnota max toku, uzaver mnoziny...**]**

[15] M(x,y,z) mapuj do Q_n. Urci n a najdi zobrazovaci funkci [x,y,z] → Q_n. Ukaz na 2 bodech a urci jejich vzdalenost v Q_n, při dil = load = 1

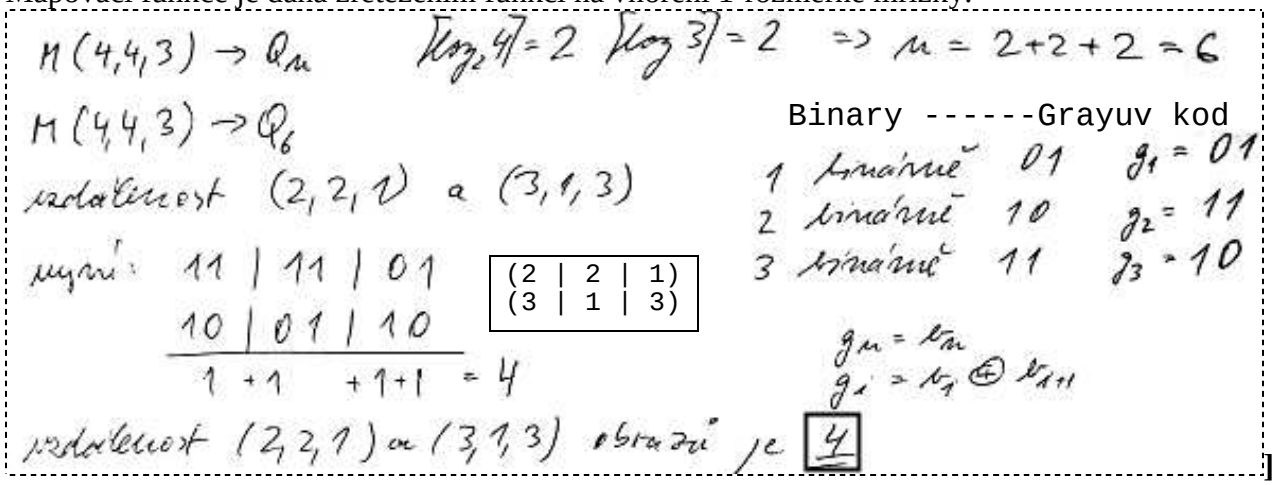
[dil = load = 1 => Mrizka lze mapovat s temito parametry do Q_n, kde $n = \lceil \log x \rceil + \lceil \log y \rceil + \lceil \log z \rceil$

Jestlize $\lceil \log x \rceil + \lceil \log y \rceil + \lceil \log z \rceil = \lceil \log(x \cdot y \cdot z) \rceil$, pak je mapovani optimalni.

1] rozlozime Mrizku na dimenze M[x], M[y], M[z]

2] provedeme Kartezsky soucin, ktery bude reprezentovat ono vnoreni

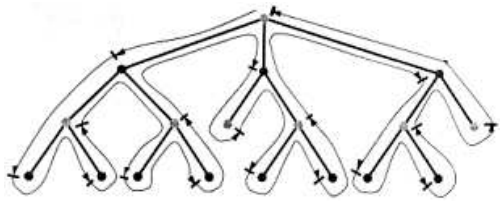
Mapovaci funkce je dana zretezenim funkci na vnoreni 1-rozmerne mrizky.



]

[16] Dokazte, ze do libovolneho souvisleho grafu G s u uzly lze vnorit T(n) s load = 1, dil ≤ 3, ecng = ?

[Konstrukcni dukaz: Postup primo vyplyva z nasledujiciho obrazku:



Uzly ***U*** libovolneho souvisleho grafu rozdelime do dvou skupin U_0 a U_1 (suda / licha uroven od korene Grafu, ktery zvolime libovolne). Vytvarime kostru grafu do hloubky a mapujeme vrcholy do kruznice(Toroidu) nasledujicim postupem:

a] uzel $U \in U_0$ vlozime do kruznice pri jeho poslednim pruchodu

b] uzel $U \in U_1$ vlozime do kruznice pri jeho prvnim pruchodu. => load=1 (kazdy uzel mapujeme pouze jednou, dil≤3 (pouze, kdyz prechazime do jineho podstromu, mame dil=3, jinak maximalne 2), ecng=2 (kazdou hranou cestujeme jednou nahoru a jednou dolu.) **]**

[17] M[11, 6, 13]. Urci počet hran a b_{WC}

[Vicerozmerna mrizka lze vytvorit z jednorozmerne kartezskym soucinem nekolika jednorozmernych. Jednorozmerna mrizka (lienarni pole) M[m] ma ***M*** uzlu a tedy (***M-1***) hran mezi nimi. Kartezskym soucinem s dalsimi dvema dimenzemi o velikostech ***n*** a ***k*** ziskame dalsich $(m-1) \cdot n \cdot k$ hran. Takto to lze udelat s kazdou hranou, vysledek je tedy celkem $\sum_{\forall i} (z_i - 1) \cdot \prod_{\substack{\forall j \\ j \neq i}} z_j$

Bisekcni sirka zalezi na velikosti nejvetsi dimenze, ***p*** v jejiz polovine provedeme rez tak, aby to bylo presne na dve poloviny a zaroven to bylo nejvyhodnejsi. Pokud je nejvetsi dimenze suda, bude b_{WC} trivialne rovna soucinu rozmeru vsech dimenzi krome nejvetsi dimenze. Pokud bude tato hodnota licha, bude se jednat o netrivialni predstavu $11 \cdot 6 + 6 + 1 = 73$ [viz obrazek] **]**

[18] Co je to e-cube smerovani na hyperkrychli a proc je odolne proti zablokovani?

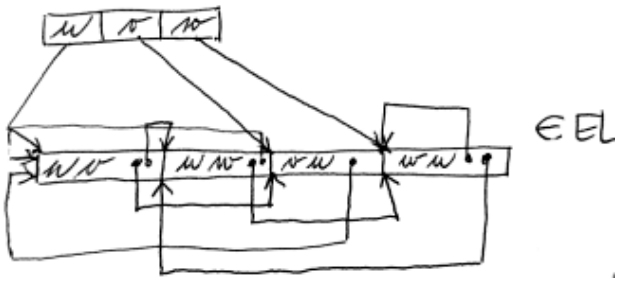
[E-cube je minimalni smerovani na hyperkrychli. Cestu mezi uzly **u** a **v** konstruujeme tak, ze porovname oba uzly nejprve od nejmene vyznamneho bitu postupne k tomu nejvyznamnejsimu. Pokud se **u** a **v** lisi v bitu ***i***, cesta se rozsiri o hranu v dimenzi ***i***. $\Rightarrow$ delka cesty je urcena poctem bitu, ve kterych se **u** a **v** vzajemne lisi a je tedy minimalni.]

[Je odolne vuci zablokovani, protože cesta se konstruuje ve smeru rostouci dimenze. Tim nemuze dojit k vytvoreni cyklu v grafu zavislosti, protože cesta blokujici hrany vyssi dimenze nemuze zadat o hrany nizi dimenze]

[19] Plnne Paralelni PRAM algoritmus pro konstruovani Eul. kruznice v n-uzlovem stromu T, PRAM s p procesory. Odvodte T(n,p) Pro standardni PRAM. Predpokladame Eul. strom, kde kazda hrana puvodniho stromu je nahrazena dvojici antiparalelnich hran.Strom reprezentujeme jako 2 seznamy:
- seznam uzlu s ukazateli do seznamu hran
- seznam hran tvoreny podseznamy hran incidujicich s jednotlivymi uzly. U kazde hrany ukazatel na hranu dvojce a podseznamy tvori cyklus

[Pouzijeme nasledujici mechanizmus, diky kteremu je mozne dosahnout konstantniho casu, pokud mame k dispozici tolik procesoru, kolik je v grafu hran. 2 antiparalelni hrany pocitame jako dve. Je zrejme, ze takovyto graf ma (2n-2) hran.

```
for vsechny_hrany_e do parallel
  ET[e] = EL[e].Dvojce->Next;
```

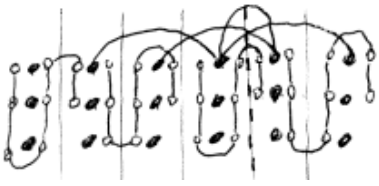


ET je pole nasledniku, EL je zretezeny seznam hran vystupujici z uzlu j. Pokud tedy mame k dispozici **p≥(2n-2)** procesoru, bude slozitost **O(1)**. Pokud mame p≤(2n-2) procesoru, dostane kazdy procesor na starosti (2n-1)/p hran.

T(n,p) = O((2n-2)/p)
C(n,p)=O(2n-2)
SU(n)=O(2n-2).**]**

```
hrana obsahuje jednak ukazatel na sve dvojce a
    jednak ukazatel na dalsi hranu z uzlu
```

[20] Urcete min y takove, ze M[8, 8] vnorime do M[3, y] s dil ≤ 2, load = 2. Urci ecng. ?



[2 sloupce dlouhe 8 zaberou 3 sloupce dlouhe 3 za predpokladu, ze load bude 2. Z toho vyplyva, ze min(y) bude 8*3/2 = 12.
ecng bude 4 = maximalni pocet starych hran prochazejicich pres jednu hranu novou. Dilatace je v tomto pripade, coz take plyne z obrazku (vzdalenost mezi jednotlivymi uzly se maximalne zdvojnasobi pri namapovani)**]**

[21] Spodni mez dilatace při vnoreni Q_{2n} do $T(2^n, 2^n)$ s load = 1 ?

[Protoze nas zajima spodni mez, budeme hledat idealni situaci. V tom pripade se nejvzdalenejsi uzly na Hyperkrychli namapuji na nejvzdalenejsi uzly na Toroidu. Prumer Q_{2n} je 2n. Prumer Toroidu je $\sum_{(i)} \frac{z_i}{2}$ pres vsechny dimenze, tedy 2^n. Vysledek tedy bude: $\frac{2^n}{2n}$.**]**

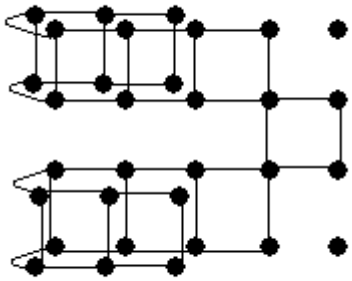
[22] CCC_n, odvodte vyraz pro prumer a kolik uzlu je v této vzdalenosti?

[Prumer bude je n negaci + (n-1) rotaci + $\lfloor n/2 \rfloor$ rotaci jeste v posledni kruznici. Jeden tah se jeste usetri pro n>3 spravnou volbou pocatecniho smeru, takze prumer Toroidu bude

$$\Phi(CCC_n) = \begin{cases} n-1+n+\frac{n}{2}-1 = 2n-2+\frac{n}{2} & ; n > 3 \\ n-1+n+\frac{n}{2} = 2n-1+\frac{n}{2} & ; \textit{jinde} \end{cases}$$ **]**

[23] Urci max y pro vnoreni M(2, y) do M(4, 5) při load = 2.

[load=2, tedy na jeden uzel mrizky (4, 5) se mohou namapovat az 2 uzly mrizky (2,y). V Jednotlivych rozich dojde k prelozeni pruhu. Max Y tedy bude rovno 16.**]**



[24] Urci max y při vnoreni M(3, y) do M(7, 9) s load = 2 a dil = 1

[Max(y) bude rovno 29 – vyplyva z obrazku. V kazdem rohu budou 3 sloupce a jinde budou 2 nebo jedna**]**

[25] CCC_6 , u = (1, 011010) v = (4, 001111), najdi minimalni cestu.

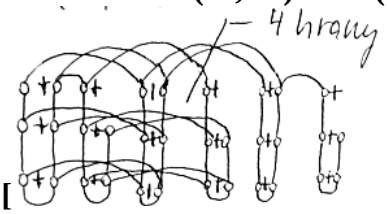
[(1,011010) => (0,011010) => (0,011011) => (1,011011) => (2,011011) => (2,011111) => (3,011111) => (4,011111) => (4,001111)
8 kroku**]**

[26] Urcete vzdalenost v SE_n grafu mezi u=1011101010 a v=1000010111. Uvazujte orientovany i neorientovany graf.

[neorient: 1011101010 → 0111010101 → 0111010100 → 1110101000 → 1101010001 → 1101010000 → 1010100001 → 0101000011 → 0101000010 → 1010000100 → 1010000101 → 0100001011 → 1000010110 → 1000010111 ⇒ 13

orient: 1011101010 → 0101110101 → 0101110100 → 0010111010 → 0001011101 → 0001011100 → 0000101110 → 0000101111 → 1000010111 ⇒ 8 **]**

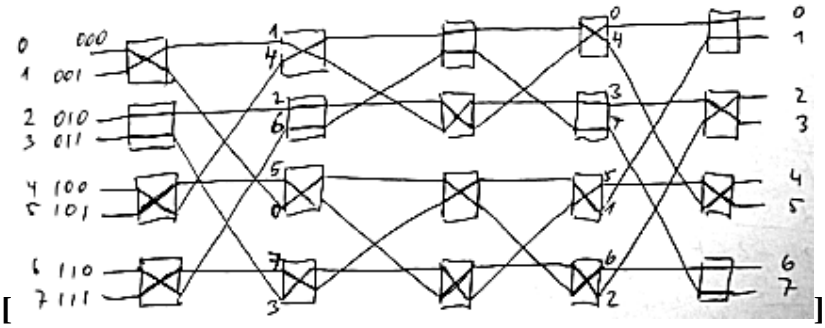
[27] Vnoreni M(11, 11) do M(3, y) při load = 2, min dil a urci ecng. ?



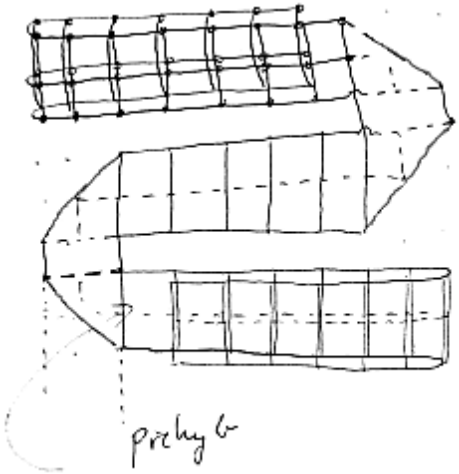
[y=22, ecng=4]

[28] Mam Benesovu sit, Jak se vytvari offline cesty pro permutace? Znazorni pro konkretni permutaci:

1 ➔ 3, 2 ➔ 0, 3 ➔ 5, 4 ➔ 4, 5 ➔ 2, 6 ➔ 7, 7 ➔ 1, 0 ➔ 6

[ **]**

[29] Urci max y, aby sla M(3, y) do M(10, 9) s load = 2 a dil = 1.

[ **y=44, ecng=2]**

[30] Urcete optimalni APRAM algoritmus pro binarni redukci, T_{OPT}, P_{OPT}, a skryte konstanty

Definice 2

operace	casova slozitost
lokalni operace	1
globalni READ nebo WRITE	d (kde $d > 1$)
k glob. READ n. WRITE za sebou	$d + k - 1$
barierova synchronizace	$B(p) = \Theta(d \log p)$

[

a] sekvencni cast: kazdy procesor secte N/p cisel a vysledek globalne zapise

b] misto klasicke binarni redukce pouzijeme n-arni redukci (***n*** bitu najednou). Sekvencni slozitost aplikace n-krat binarni operace ® je priblizne SU(n)=2N

$T_R(n, p_P) = <\mathsf{RLW}> = 3\tau_P \Rightarrow C(n, p_P) = p_P \cdot 3\tau_P$

$T_a(n, p_P) = <\mathsf{RBLWB}> = \big(d + B(p) + 1 + d + B(p)\big)\tau_P = \big(2B(p_a) + 2d + 1\big)\tau_P = \Theta(B(p_a)\tau_P)$

cena je tedy o poznani horsi nez u PRAM reseni. Jak snizit cenu jsou 2 moznosti. Zmensit dobu potrebnou pro jeden cyklus <RBLWB>, nebo zmensit pocet tech cyklu. Prvni moznost je vyloucena, protoze bariery, ktere prave maji tak spatnou slozitost odstranit nemuzeme kvuli hazardum. pouzijeme tedy nasledujici uvahu:

$$p_b = \frac{p_a}{B(p_a)} = \frac{p_p}{B(p_p)} \qquad <\underbrace{\mathsf{R}}_{B(p_p)}.\underbrace{\mathsf{RBL}}_{B(p_p)}.\underbrace{\mathsf{LW}}_{B(p_p)}.\mathsf{WB}>^{\frac{t}{B(p)}}$$

$$T_b(n, p') = \frac{t}{B(p)}\big(d + B(p) - 1 + B(p') + B(p) + d + B(p) - 1 + B(p')\big) = t_A\big(3B(p) + 2B(p') + 2d - 2\big)$$

$$B(p') = d\log p' = d\log\frac{p}{B(p)} = d\left(\log p - \log(d\log p)\right) = d\log p - d\log d - d\log\log p \approx d\log p = B(p)$$

$$T_b(n,p') = t_A\left(5B(p) + 2d - 2\right) \approx 5t_A d\log p$$

$$C_b(n,p') = \frac{p}{B(p)}5t_A d\log p \le 5pt \qquad \text{(pro srovnani PRAM mel: } C(n,p) = 3pt\ \text{)}$$ **]**

[31] M(14, 16, 10) , souradnice od 0, najdi všechny minimalni uzlove disjunktni cesty mezi u = (12, 4, 8) a v = (5, 9, 3). Reseni vysvetli, popis algebraicky a schematicky naznac

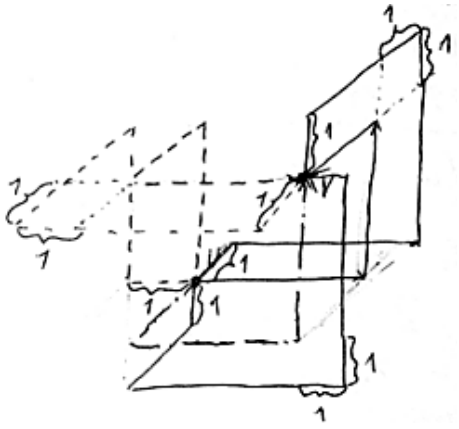
[a] nejkratsi cesty ziskam takto: pomoci XYZ smerovani dostanu uplne nejkratsi a pak rotuju (YZX, a ZXY)

b] dalsi co mozno nejkratsi disjunktni cesty ziskam z tech ziskanych v a] + jim prodlouzim cesty o 4 jednotky nasledovne:

(dohromady tedy budu mit tolik disjunktnich cest, kolik je stupen uzlu v 3D Mrizce)

1) posunu se v dimenzi o jedna mensi nez original z bodu a] opacnym smerem, nez pravy algoritmus pro nejkratsi cestu (vzdalim se o 1 od ciloveho uzlu v dane dimenzi).

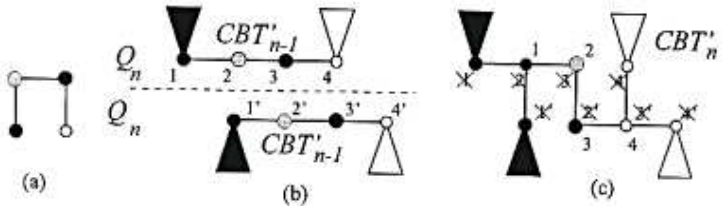
2) nyni udelam vsechno stejne jako v a] az do chvile, kdy se chci posunout v dimenzi, ve ktere jsem se posouval v bode b] 1). Misto tohoto pohybu jeste o jedna setrvam ve stavajici dimenzi a pak se teprve presunu do dimenze, ktera mela nasledovat. Nejprve se posunu o 1 a dotvorim tzv. roh. Pak zase pokracuju podle originalu. Na konci se posunu opacnym smerem nez byl uplne prvni krok v b] 1). Tim se mi cela cesta posunula o 4 tahy. **]**



pro trojdimenzionalni mrizku bude 6 ruznych disjunktnich cest, protoze kazdy uzel ma 6 sousedu

[32] Popis algoritmus pro vnoreni CBT_n do Q_{n+1} s load = ecng = 1, dil = 2. Zkonstuuj a popis CBT_3 do Q_4.

[Kvuli ruzne topologii obou grafu neni mozne vlozit CBT_n do Q_{n+1}, s dil=1, coz vsak neni predmetem teto ulohy. Zakladnim elementem tohoto prikladu je otazka, jak se vnori CBT_1 do Q_2. Postup je takovy, ze se "*zdvoji koren*" CBT a vsechny uzly se namapuji podle obrazku. Pote se jiz pouze pouzije rekurentni sablona na vkladani Uplnych binarnich stromu (CBT) do Hyperkrychli o jednu dimenzi vetsich.

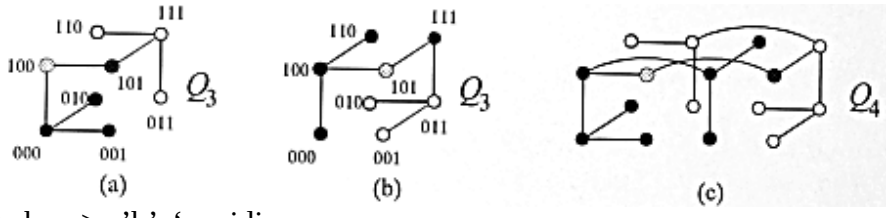


Nakonec bude potreba nejak vyjadrit mapovaci funkci. Napiseme si nasledujici posloupnost promitnuti a z ni danou funkci vycteme:

1=>1‘: 000 => 100
2=>2‘: 100 => 101
3=>3‘: 101 => 111
4=>4‘: 111 => 011

nyni si oznacime jednotlive sloupce binarnich cislic(dimenze) abc => a’b’c‘ a vidime, ze:

f(a b c)=not(b) c a=a’b’c‘



]

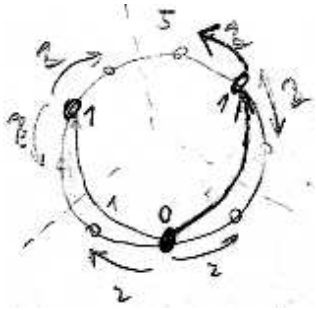
[33] Dokazte, ze e-cube smerovani na Q_n je pro permutaci zhusteni bezkolizni.

[Uvazujme nejprve prvni dimenzi (dimenze nejnizsiho bitu). Mohou nastat 2 pripady:
1] posledni bit je pro oba uzly stejny => ke kolizi nedojde, protoze v e-cube se nebude odehravat zadny pohyb
2] posledni bit je ruzny => e-cube smerovani bude generovat hranu, ktera bude v teto nejnizsi dimenzi. Oba uzly se pohnou a vymeni si pozici – nedojde tedy ke kolizi.
Timto krokem se kazdy z uzlu vnori do jine podkrychle Q_{n-1}
Rekursivnim postupem dokazeme totez pro tyto 2 podkrychle.**]**

[34] Dokazte, ze e-cube smerovani na Q_n je pro operaci prelozeni $i \rightarrow i \oplus w$ bezkolizni.

[E-cube se ridi bity retezce $w = u \oplus v$. Mohou nastat 2 pripady:
1] k-ty bit retezce w je 0 => neco $\oplus$ 0 je neco => nedojde k zadnemu pohybu => ani ke kolizi
2] k-ty bit retezce w je 1 => neco $\oplus$ 1 je neg(neco) => dojde k pohybu u obou uzlu (jak u tak v), dojde tedy k vymene pozice a nedojde tedy ke kolizi**]**

[35] Nasobeni matice [m x n] vektorem [n x 1] na T($\sqrt{p}$, $\sqrt{p}$), WH, vseportovy, mapovani blokove sachovnicove, vstupni vektor na poslední sloupec.



[(n x n) * (n x 1) na T($p^{1/2}$, $p^{1/2}$) vseport.
Pouzijeme nasledujici postup:
a] procesor v poslednim sloupci posle svoji cast vektoru na diagonalu
b] procesory na diagonale udelaji OAB a rozeslou $\boldsymbol{x_i}$
c] kazdy $\boldsymbol{p_{[ij]}}$ spocita svoji lokalni cast
d] provedeme paralelni redukci na jednotlive radky s binarni operaci scitani. Korenem redukce budou uzly napravo v poslednim sloupci prumer Toroidu je $p^{1/2}$

```
Po prvnim kroku jsou informovany 3
  uzly. Kazdy si vezme na starost
        jednu tretinu prace
```

$$T_a = t_s + \frac{\sqrt{p}}{2} t_d + \frac{n}{\sqrt{p}} t_m$$

$$T_b = t_s + \frac{n}{\sqrt{p}} t_m \log_3 \sqrt{p} + \frac{\sqrt{p}-1}{2} t_d$$

$$T_c = \left| \text{kazdy udela } \frac{n}{\sqrt{p}} \text{ soucinu vektoru dlouhych } \frac{n}{\sqrt{p}} \right| =$$

$$= \frac{n^2}{p}$$

$$T_d = \left| \begin{array}{l} \text{vzdalenost mezi komunikucicimi procesory se} \\ \text{zvetsuje - na WH to ma maly vliv} \end{array} \right| =$$

$$= t_s + \frac{n}{\sqrt{p}} t_m \log_3 \sqrt{p} + \frac{\sqrt{p}-1}{2} t_d$$

$$T(n,p) = O\left(\frac{n}{\sqrt{p}} \log_3 p + \frac{n^2}{p}\right) = \left(\frac{\sqrt{N}}{\sqrt{p}} \log p + \frac{N}{p}\right)$$

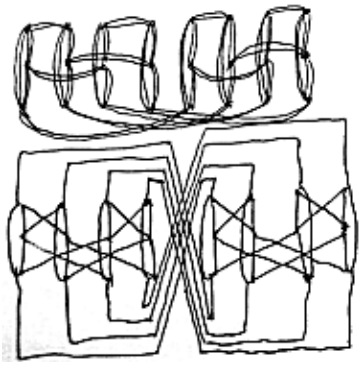
[36] Dany vnitrni uzly mrizky u, v ve 4D mrizce. Zkonstruuj všechny nejkratsi uzlove disjunktni cesty z u do v. Reseni vysvetlete a popiste algebraicky, doplnte schematickym nacrtkem. Urci delky a pocty cest v k-rozmerne mrizce.

[viz příklad 30. Nazorny obrazek pujde naznacit napriklad takto:**]**

[37] Dokazte, ze CCC_n a wBF_n jsou kvaziizometricke, nakreslete pro n = 3.

[Kvaziizometricke ➔ navzajem lze vnorit s konstantnimi parametry vnoreni.
a] CCC_n do wBF vlozime pouzitim mapovaci funkce f(i, x) = (i ⊕ parita (x), x)
b] wBF do CCC_n vlozime pomoci zobrazeni *identita* s parametry load = 1, dil = ecng =2

1 hyperkubicka hrana z wBF_n se mapuje na jednu hyperkubickou hranu z CCC_n a jednu kruznici z CCC_n.]



[38] Mame oBF_n , SF prepinani, monotonni permutace. Dokazte, ze kazdou monotonni permutaci lze na SF oBF_n realizovat bezkolizne, to je v O(n) krocich. Odhadnete skrytou konstantu.

[Libovolnou monotonni permutaci lze na motylku realizovat jako spojeni:
1] zhusteni
2] roztahovani na prevracenem motylku
3] identita
add 1 – Identita je trivialne bezkolizni
add 2 – Roztahovani na prevracenem motylku je zrcadlova vuci zhusteni
add 3 – Kolize muze nastat pouze mezi sudo-lichou dvojici paketu. Na vystupu se po sobe objevi jako dvojice
a] sudo-licha => oba pakety jdou rovne => kolize neni
b] licho-suda => nejnizsi bit se meni => oba jdou krizem => kolize neni
Po tomto kroku se dostanou na vstup podmotylku (kazdy do jineho) Jejich dalsi cesta je tedy disjunktni. Nyni se lze vratit na zacatek do bodu add3 a aplikovat pro jednotlive podmotylky cely rekurzivni postup. Vsechny 3 operace jsou tedy bezkolizni. Jejich slozitost je O(n). Celkovy pocet kroku je tedy mensi nez 3n => skryta konstanta je 3. **]**

[39] Uved casove a cenove optimalni algoritmus pro vypocet RANK nad seznamem reprezentovanym pomoci pole ukazatelu Succ na CREW PRAM. Zhodnot skalovatelnost.

[Sekvencni reseni je trivialni. Staci pro nej projit jednou seznamem a mame hotovo. Paralelni reseni se provadi metodou, ktera se nazyva: preskakovani ukazatelu (pointer jumping). Pro jednoduchost budu predpokladat, ze p=N

```
for i = 1 .. n do_parallel
  if Succ[i] == i  Rank[i] = 0 else Rank[i] = 1
  for ( k=0; k < logN; k++ ) Rank[i] += Rank[Succ[i]]; Succ[i] = Succ[Succ[i]]
end.
```

Pro p<N bude kazdy procesor simulovat N/p procesoru. Primo z algoritmu je videt, ze:
T(N,N) = konst * log (N) = O(log N)
tedy T(N, p) = O(N/p * logN), C(N, p) = T(N, p) * p = O(N * log N)
Problem je, ze neexistuje lokalita dat. vysledkem lokalniho preskakovani nad N/p prvky je opet N/p prvku => nedochazi k redukci mnozstvi dat. Navic stale pristupujeme k poslednimu prvku(prvnimu), takze prace take neni prizniva.**]**

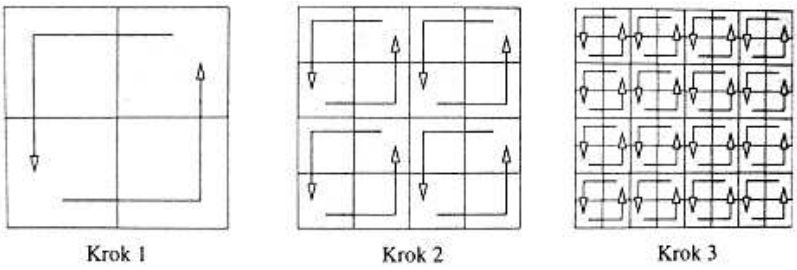
[40] Popiste algoritmus pro transpozici matic [n x n] na p procesorech v Q_n s WH, je-li A map. sachovnicove – blokove. Odvodte počet komunikacnich kroku a casovou slozitost

[

Namapovani mrizky M na krychli Q je jednoduche – [priklad 15].

$T(n^2, p) = \left(t_s + 2\frac{n^2}{p} t_m \right) \frac{q}{2} + \mathrm{O}\left(\frac{n^2}{p} \right)$, WH/SF

prepinani nehraje prilis roli, protoze vsechny cesty maji delku 2. Bohate staci poloduplexni SF jednoportova, protoze hrany jsou vzdy k dispozici 2(2 cesty)**]**



[41] Urcete velikost všech podstromu. Je zadano pole RANK[ ], EL[ ], AL[ ]

[

```
for vsechny_hrany e do_parallel
  if Rank[e]<Rank[EL[e].Dvojce] DIR='F'
    else DIR='R'
  for vsechny_hrany xy do_parallel
    if DIR=='F' Parent[y]=x
  Parent[Root]=nil;
  for vsechny_uzly u && u != Root do_parallel
    v=Parent[u]
    Subtree[u]=(Rank[uv] - Rank[vu] + 1) / 2
Subtree[Root]=n
```

Pocet procesoru nutnych k realizaci algoritmu v konstantnim case je roven poctu hran = $2n-2$. Pri mensim poctu procesoru je $T(n, p) = \mathrm{O}\left(\frac{2n-2}{p} \right)$. Cena je rovna cene sekv. algoritmu $\Rightarrow$ jedna se o plnne paralelni algoritmus.**]**

[42] PPS na 1D jednoportove mrizce. Jak se zmeni pro 2portovou mrizku?

[Jednoportova 1D Mrizka nedovoluje lepsi reseni nez je sekvencni. Casova slozitost bude linearni, pokud pocet procesoru odpovida velikosti reseneho problemu. Pro

$p < n$ je $T(n, p) = \frac{n}{p} + p + 1 + \frac{n}{p} - 1$.

2portova mrizka nic neresi.**]**

[43] AAB na mrizce, SF, urci t_{AAB} a r_{AAB}

[44] Vnoreni M(22, 24) do M(x, 5) s load = 2, dil = min, urci ecng

[45] Permutace prohozeni na motylku.. Proc je kolizni? Jak to udelat lepe?

[Je mozne nahradit kolizni prohozeni za nasledujici postup. Setridime pakety na vstupech motylka podle cilove adresy. Je-li ziskana posloupnost uplnou permutaci, provedeme operaci *identita*
Neni-li permutace uplna, musime nejprve udelat roztazeni a pote identitu. Casova slozitost bude dana casobou slozitosti setrideni.**]**

[46] PPS na EREW PRAMu. Odvodte spravnost, spocti efektivnost a skalovatelnost.

[47] **Transpozice matice [n x n] na M($\sqrt{p}$, $\sqrt{p}$), WH, XY, vseportova**

[48] **OAS na T(z_1, z_2), 1-port, WH, kombinujici. Urci [ro], [tau], r, t. Porovnej optimalitu.**

[49] **Multicast na mrizce, smerovani WH, 1-portova, alg. ukaz na M(5, 5)**

[50] **Urci dolni mez na počet kroku AAS v mrizce, nekomb.**

[51] **OAS na 1-port WH na T(z_1, z_2). Urci r_{OAS}, t_{OAS}**

[52] **Popis EREW PRAM algoritmus pro vypocet poradi uzlu preoder všech uzlu. Euleruv strom, n uzlu, 2n-2 = m hran, reprezentovan pomoci pole uzlu AL, pole hran EL a reprezentace Eulerovske cesty EA (pole). Odvodte skalovatelnost.**

[53] **Popiste krokove a casove optimalni OAB na 2-portove 1D mrizce M(z) s WH prepinanim. Vysilac je v levem uzlu site, urcete r_{OAB} a t_{OAB}.**

[54] **Popiste polylogaritmicky slozity algoritmus simulace prioritniho CRCW na EREW s pametovou slozitosti O(m + p). Počet procesoru na EREW p‘ = max (m‘ , p), m‘ = m. Navíc potrebuju pomocne pole A o p bunkach. Rozlisime operace READ, WRITE a LOCAL COMP.**

[55] **Algoritmus binarni vymeny (BEX) na mrzice M($\sqrt{p}$, $\sqrt{p}$), 1-portova, plnne duplexni, WH prepinani, kombinujici, urci t_{AAB}, r_{AAB}.**

[56] **Bitonic Sort na $Q_{\log p}$, $1 \leq p \leq N$. Urci T(n, p), φ_1, φ_2, zhodnotte skalovatelnost, lze udrzet konstantni efektivitu pro p = O($\sqrt{N}$) ?**

[57] **2-portovy toroid T(z), OAS, WH prepinani, urci t_{OAS}, p_{OAS}.**

[58] **3D Sort – Skalovatelnost**

[59] **EREW PRAM alg. cislovani uzlu postorder. Predpokladame Eul. strom a jeho reprezentaci polem, mam hotovu eulerovu cestu.**

[60] **Popiste Cannonuv algoritmus na T(z, z), plnne duplexni, 2-portova, WH. Urcete spodni mez na počet kroku a casovou slozitost.**

[61] Multicast na Q_n, WH, 1-portova

[62] Napiste optimalni algoritmus pro OAB na 2-portove mrizce M(x, y), full duplex, WH prepinani, Urci spodni mez poctu kroku , casovou slozitost.

[63] Popiste algoritmus sudo-liche redukce pro reseni tridiagonalni soustavy rovnic n, mapovanych blokove radkove na p-procesorovou hyperkrychli, prepinani SF, $t = t_S + dt_m\mu$, odvodte T(n, p)

[64] … jako minule, jen Q_n je WH => nedojde prakticky k zadne zmene, protože predchozi model vhodneho mapovani procesoru na Q_n tak, ze procesor p_i namapoval na uzel Q_n odpovidajici retezci u_2 = BRGC(binary(i))
Tim se jakakoliv komunikace odehrava po max. dlouhe ceste 2.

[65] Shearsort, N cisel na M($\sqrt{p}$,$\sqrt{p}$). Odvod T(n, p), zhodnot skalovatelnost a urci T_{MIN} a spodni mez poctu procesoru pro dosazeni opt. φ_3.

[66] Popiste jednu iteraci jakehokoliv alg. a urcete co nejpresneji slozitost této iterace na 1D mrizce M(z)

[67] PPS na Q_n, WH, casy, skalovatelnost

[68] Odvodte spodni mez na počet kroku a cas z OAS, 1-portova, WH 2D mrizka, zdroj na [a_1, a_2], popiste efektivni algoritmus a porovnejte se spodni mezi. S kombinovanim zprav?

[69] Popiste opt. alg. pro AAS na $Q_{\log p}$, WH, full duplex, bez kombinovani. Urci [tau], t, r a efektivnost.

[70] QEX na M($\sqrt{p}$,$\sqrt{p}$), AAS, WH. [tau] = ?. Omezeni bisekcni sirkou: $b_{WC} = \sqrt{p}$

[71] FOX na $Q_{\log p}$, urcit φ_1, φ_2, pametovou slozitost, WH. Mam 2 matice A,B (n x n). Obe jsou rozdeleny blokove sachovnicove na procesy stejne, jako kdyby se jednalo o mrizku ➔ vnorime virtualni toroid do hyperkrychle Q_r, p = 2^r. Pocatecni namapovani submatic o velikosti n2/p ukazuje obrazek:..

[72] Urcete φ_3(n), jestlize 1≤p≤n a: $T(n, p) = \frac{2n}{p} + 6\log p$

[Postup je nasledujici:
1) najdi z **T(n,p)** takovy pocet procesoru **p**, aby cas byl minimalni (derivace)
2) dosad tento pocet procesoru **p** do **T(n,p)** a ziskej minimalni cas T_{min}
3) Poloz puvodni **T(n,p)** rovnu radove minimalnimu casu ziskanemu v bode 2)

3) Vyjadri rad pro **p**. Ziskany vyraz pro **p** je roven hledane funkci

$$T(n, p) = \frac{2n}{p} + 6\log p$$

1) $\frac{\partial T}{\partial p} = 0 \quad \Rightarrow \quad -\frac{2n}{p^2} + \frac{6}{p \ln 2} = 0 \qquad \Leftrightarrow \qquad p_{opt} = \frac{\ln 2}{3} n$ **]**

2) $T_{\min} = 6\log n - 4$

3) $\frac{2n}{p} + 6\log p = \mathrm{O}(\log n) \qquad \Rightarrow \qquad p = \varphi_3(n) = \frac{n}{\log n}$

[73]