**Лекция № 12.** Параллель алгоритмдер. Полиномды және экспонентталық алгоритмдер. Регистр саны шектеулі машина.

**МАҚСАТЫ:** Параллель алгоритмдер,полиномды және экспонентталық алгоритмдер, регистр саны шектеулі машина туралы ұғымдармен таныстыру.

**СҰРАҚТАР**:

1. Параллель алгоритмдер.
2. Полиномды және экспонентталық алгоритмдер.
3. Регистр саны шектеулі машина.

**НЕГІЗГІ ҰҒЫМДАР:** Параллель алгоритмдер. Полиномды және экспонентталық алгоритмдер.Регистр саны шектеулі машина.

Қазіргі кезде көпядролы микропроцессорлардың шығуына байланысты компьютердің аппараттық құралдарын тиімді пайдалану ретінде парраллель алгоритмдерді қолдануды түсінеді.

Параллель жүйелердегі тиімді программаларды жасау 3 негізгі компоненттен тұрады: параллель алгоритмдер, параллелдікті іске асыру құралдары, жөндеу (отладка) жүйесі.

Параллелдікті іске асыру құралдары деп параллел программалардың инфрақұрылымын құрайтын программалау тілдерін немесе кластар кітапханасын түсінеді. Мұндай жүйелер көп. Оларға [Occam](http://wotug.kent.ac.uk/parallel/occam/" \t "_blank), MPI, [HPF](http://www.netlib.org/hpf/" \t "_blank), [OpenMP](http://openmp.org/wp/" \t "_blank), [DVM](http://www.keldysh.ru/dvm/" \t "_blank), [OpenTS](http://www.opents.net/index.php/en" \t "_blank), [Boost.Thread](http://www.boost.org/doc/libs/1_37_0/doc/html/thread.html" \t "_blank), [Posix Threads және](https://computing.llnl.gov/tutorials/pthreads/" \t "_blank) Intel компаниясының [Integrated Performance Primitives кітапханасын жатқызады.](http://software.intel.com/en-us/articles/intel-ipp/" \t "_blank)

Жай тізбекті программаларға қарағанда параллель алгоритмді жөндеу (отладка) қиын процесс болғандықтан, мұндай жүйелерде жөндеу және профильдеу жүйелері маңызды бөлік болып табылады. Жөндеу жүйелеріне TotalView және PGDBG атап өтуге болады. Ал профильдеу жүйелеріне Nupshot, Pablo, Vampir саймандарын жатқызады. VivaMP сияқты статикалық верификация жүйелері де бар. Мұндай саймандар ретінде [Threading Analysis Tools көпағындылықты талдау жүйесін де атауға болады..](http://www.intel.com/cd/software/products/asmo-na/eng/219785.htm)

Алайда, жүйедегі басқа компоненттері программаны параллель ете алмайтын, онсыз жұмыс істей алмайтын ең басты компонент − бұл параллель алгоритм.

[Информатикада](http://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)  параллель алгоритм дәстүрлі тізбекті алгоритмге қарама қарсы қойылады, бұл алгоритм әртүрлі көптеген есептеу құрылғыларында бөлік бойынша жүзеге асырылып, соңынан алынған нәтижелерді біріктіруден және дұрыс нәтиже алудан тұрады.

Кейбір алгоритмдер оңай тәуелсіз орындалатын фрагменттерге бөліне алады. Мысалы, распределение работы по проверке всех чисел от 1 − 100000 аралығындағы сандарды жай сандарға тексеру жұмыстарын үлестіріп беру әрбір процессорға осы сандар жиынын бөліп беріп, соңынан алынған нәтижелерді − жай сандар жиындарын біріктіруден тұрады, осылай [GIMPS](http://ru.wikipedia.org/wiki/GIMPS) жобасы іске асқан.

Алайда [пи](http://ru.wikipedia.org/wiki/%D0%9F%D0%B8) \left( \pi \right)  мәнін есептейтін көптеген танымал алгоритмдерде параллель орындалатын бөліктерге бөлу мүмкін емес, өйткені келесіні есептеу үшін алдыңғы нәтиже қажет болады. Итеративті сандық әдістер, мысалы,  [Ньютон](http://ru.wikipedia.org/wiki/%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0) әдісі, үш дене есебі таза тізбекті алгоритмдерге жатады. Кейбір  [рекурсивті](http://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%BA%D1%83%D1%80%D1%81%D0%B8%D1%8F" \o "Рекурсия) алгоритмдер де параллель орындатуға келмейді. Мысалы графтағы тереңдікке бойлай іздеу алоритмдері соған жатады.

Параллель алгоритмдер көппроцессорлы жүйелерді дамыту үшін, өазіргі процессорлардағы ядролар санын арттыру үшін аса маңызды. Әдетте, өгімділіктері бірдей көптеген баяу жұмыс істейтін процессорлары бар компьютерге қарағанда, бір жылдам процессоры бар компьютерді құрастыру оңай. Алайда процссордың өнімділігі техпроцессорды жетілдіру арқылы іске асады, бұған микросхема элементтерінің размеріне және жылу бөлінуіне қойылатын шектеулер әсер етеді. Бұл шектеулерді көппроцессорлы өңдеу арқылы шешуге болады., бұл тіпті шағын есептеу жүйелеіне де тиімді болып шықты.

**Параллель алгоритмдердің күрделілігі мынада:** алгоритмді орындау үшін пайдаланатын жад көлемі мен уақытта (процессор такттерінің саны). Параллель алгоритмдер тағы бір ресурсты ескеруді талап етеді: түрлі процессорлар арасындағы байланыстардың ішкі жүйелері. Процессорлар арасында алмасудың екі түрі бар: ортақ жадты пайдалану және хабар жіберу жүйелері.

Ортақ жад жүйелері қосымша процессорларды қолдануға нақты шектеулер қойып, өңделетін деректерге қосымша бұғаттауды қолдануды талап етеді.

Хабар жіберу жүйелері каналдар ұғымын және хабарлар блоктарын қолданады, бұл шинада қосымша трафикті, хабарлардың кезектерін ұйымдастыру үшін жадтың қосымша жүмсалуын қажет етеді. Қазіргі процессорлардың дизайнында есепті орындау кезінде хабар алмасудың әсерін азайту үшін арнайы коммутаторлар (кроссбарлар) қарастырылған.

Параллель алгоритмдерді қолдануға байланысты тағы бір проблема − бұл баланстау, Мысалы, 1-100000аралығындағы жай сандарды іздеуді қолда бар процессорлар арасында оңай бөлуге болады, алайда кейбір процессорлардың жұмысы көп болып, кейбіреуі өңдеуді жылдам аяқтап, қалғандары жұмысын бітіргенша бос тұрып қалуы мүмкін. Жүктемені баланстау проблемасы гетерогенді есептеу орталарын қолданғанда тіпті артады, олардағы есептеу элементтері өнімділігі мен қолжетімділіг жағынан айтарлықтайқатты өзгешеленеді (мысалы,  [грид](http://ru.wikipedia.org/wiki/%D0%93%D1%80%D0%B8%D0%B4" \o "Грид)-жүйелерде).

Үлестірілген алгрритмдер деп аталатын параллель алгоритмдердің бір түрі кластерлерде және үлестірілген есептеу орталарында қолдану үшін үлестірілген өңдеу ерекшелігін ескеріп арнайы жасалады.

**Параллель алгоритмдерді мына есептерді шешуде қолдану мысалдары**: матрицаны матрицаға көбейту, Дирихле есебі, Гаусс әдісімен және қарапайым итерация әдісімен сызықты теңдеулер жүйесін шешу (СЛАУ), n-өлшемді векторларды скаляр көбейту алгоритмі. Бэтчердің сұрыптау алгоритмі аса тиімді емес, алайда оның басты артықшылығы: барлық салыстырулар мен орын ауыстыруларды бір мезгілде орындатуға болады. Алгебралық және трансценденттік теңдеулердің түбірлерін табу және т.б. есептер.

Экспоненциальды және полиномды алгоритмдер.

Экспоненциальды күрделілік немесе  экспоненциальды уақыт — алгоритмдер күрделілігі [теорииясында е есепті шешуге кеткен уақыт,  ол есептің өлшеміне байланысты](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9" \o "Теория сложности вычислений) [[экспонентамен](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9" \o "Теория сложности вычислений)](http://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B0) [шектелген. Басқаша айтсақ,](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9" \o "Теория сложности вычислений) егер есептің өлшемі сызықты артса, оны шешу уақыты экспоненциальды түрде артады.

Экспоненциальды және полиномды алгоритмдердің айырмашылықтары  [фон Нейман](http://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%B9%D0%BC%D0%B0%D0%BD,_%D0%94%D0%B6%D0%BE%D0%BD_%D1%84%D0%BE%D0%BD) бойынша артады.

**Полиномды күрделілігі бар алгоритмдер** «жылдам», ал күрделілігі полиномды алгоритмдерден артық алгоритмдер «баяу» деп саналады. Осы тұрғыдан экспоненциальды алгоритмдер «баяу» деп саналады. Алайда бұл дұрыс емес. Өйткені алгоритм жұмысының уақыты есептің өлшеміне (n) және [O-нотаци](http://ru.wikipedia.org/wiki/O-%D0%BD%D0%BE%D1%82%D0%B0%D1%86%D0%B8%D1%8F)яда жасырылған константаға тәуелді.

Кейбір n-нің аз мәні үшін экспоненциальды уақыттан да артық болуы мүмкін. Дегенмен, n-нің көптеген мәні үшін экспоненциальды күрделіліг бар алгоритмдердің жұмыс уақыты айтарлықтай үлкен болады.

Полиномды уақыттан артық(«сверх-полиномиальное») , бірақ экспоненциальды уақыттан кем («суб-экспоненциальное») жұмыс істейтін алгоритмдер кездеседі. Мысалы, бұған бүтін санды жай көбейткіштерге жіктеу ([факторизация](http://ru.wikipedia.org/wiki/%D0%A4%D0%B0%D0%BA%D1%82%D0%BE%D1%80%D0%B8%D0%B7%D0%B0%D1%86%D0%B8%D1%8F)) есебі жатады. Мұндай «баяу» алгоритмдерге жатады.

Экспоненциалды алгоритмдерді іздеу жөнінде сөз қозғағанда, жиі ықтимал әдістер пайдалы болып келеді. Кей жағдайларда есептеулер аяқталмауы немесе тіпті дұрыс емес жауап беруіне жол беріледі. Дегенмен, кейбір кезде олардың практикалық қолданысқа ие болуы мүмкін.

*Т*(*п)* уақытында жұмыс істейтін және *p* ықтималдылығымен дұрыс жауап беретін және тек қана бір жаққа қателесетін алгоритм белгілі болсын. Егер оны *■ p-1* рет қайталайтын болса, онда нәтижесінде қателік ықтималдылығы келесідегідей болады:

(1-p)lP <е-°

Константтар ескерілмеген соң, кез келген алдын ала берілген константадан кем болатын ықтимал қателікті сұрауға болады. Мысалы, есептеу барысында жаңылысу ықтималдылығы сияқты. Сол себепті тәжірибе жүзінде ықтимал алгоритмдер детерминирленген сияқты кең қолданыста болады.

**Полиномиалды есептерге мысал.**Қарапайым идея бояу туралы есеп үшін ықтимал алгоритмді алуды мүмкін еткізеді. Ол одан да қарапайым есеп үшін полиномиалды шешімді қолданады. Кездейсоқ түрде әрбір төбе үшін оны бояй алатын бірнеше түстер белгілейік. Әрбір төбе үшін сәйкес келетін жұбын табу 2*/*3 құрайды, ал онда осындай «алдын ала бояуды» табу ықтималдылығы әрбір төбе үшін (2*/*3) болып келеді. Ал егер әрбір төбе үшін екі түстің бірін таңдау қажет болса, онда полиномиалды алгоритм қолдануға болады.

Сөйтіп, бірнеше рет осындай операцияны қайталау жұмыс уақытын O(1.5")-мен бағалаудың ықтималды алгоритмін береді. Сонымен бірге қате ,тек теріс нәтиже орын алуы мүмкін.

**Көпмүшелердің мәнін есептеу**

Келесі түрдегі жалпы көпмүшемен айналысамыз

р ( х ) = апхп + an-ixn~l + ап~2Хп~2 Н Н а2х2 + а\х + а0. (4.1)

**Есептелген алгоритмдер**

Біз барлық а коэффициенттерін болжаймыз An,..., A0 белгілі, тұрақты және массивке жазылған. Бұл дегеніміз, бірден бір кіріс деректері көпмүшені есептеу үшін мән болып табылады, ал бағдарламаның нәтижесі көпмүшенің х нүктесіндегі мәні болуы керек.

Сызықты есептеудің стандартты алгоритмі:

Evaluate(х)

х нүктесі көпмүшелердің мәнін есептейді.

result=a[0]+a[l]\*x

xPower=x

for i=2 to n do

xPower=xPower\*x

result=result+a[i]\*xPower

end for

return result

Бұл алгоритм анық және оның талдауы нақты.

for циклінде *п —* 1 рет орындалатын екі көбейту бар.

Одан басқа, бір көбейту циклден бұрын орындалады, сондықтан көбейтудің жалпы саны тең: *2п —* 1. Циклде сонымен қатар бір қосу амалы да болады, және бір қосу циклдің алдында орындалады, өйткені қосудың жалпы саны тең болады: *п.*

**Горнер схемасы**

Горнер схемасы бойынша есептеу анағұрлым тиімді, өйткені ол өте күрделі емес. Бұл схема келесіде көрсетілген көпмүшеге негізделген:

р(х) = ((... ((апх + an-i)x + а„\_2)ж -I + а2)ж + а\)х + а0.

(4.2) Оқырмандар тез арада тексере алады, бұл көрсетілімнің

(4.1) өрнегіндегімен бірдей екенін. Сәйкес алгоритмдер келесідей бейнеде болады:

HornersMethod(x)

х нүктесі көпмүшелердің мәнін есептейді.

for i=n-l down to 0 do

result=result\*x

result=result+a[i]

end for

return result

Цикл *п* рет орындалады, өйткені циклдің ішінде бір көбейту және бір қосу бар. Сондықтан Горнер схемасы арқылы есептеу кезінде *п көбейту және п қосу орындалады* — бұл стандартты алгоритммен салыстырғанда көбейту саны екі есе азаяды.

Коэффициенттердің болжамды өңделуі

Нәтижені жақсартуға болады, егер көпмүшелердің коэффициенттерін өңдеу алгоритм жұмысының басталуына дейін болса.

Басты идея мынада, көпмүшелерді аз дәрежедегі көпмүшелер арқылы көрсету. Мысалы, 256 –ді есептеу үшін сондай циклді пайдалануға болады. Evaluate функциясы сияқты. Нәтижесінде 255 көбейту орындалады. Альтернативті әдіс былай құралады:

result=x\*x өрнегін қоямыз, ал содан келесі операцияны жеті рет қайталаймыз: result=result\*result. Бірінші result айнымалысы орындалғаннан кейін ж4 құрамға енеді, екінші ж8-ден соң, үшінші ж16-дан соң, төртінші ж32-дан соң, бесінші ж64, алтыншы ж128, және жетінші ж256-дан соң.

Болжау бойынша коэффициенттерді оңдеу жұмысы жақсы жүруі үшін, көпмүшелер унимодальды болуы керек (яғни үлкен коэффициент ап бірге тең болуы керек), ал көпмүшенің дәрежесі кейбір екілік дәрежелерден біреуге кем болуы керек (п = 2 — 1 кейбіреулері үшін: k > I). Мұндай жағдайда көпмүшелер келесі түрде болады:

р(х) = (xj + b)q(x) +г(ж), rflej = 2fc~~1. (4.3)

Екі көпмүшеде де q және г кіші мүшелі болады, р-мен салыстырғанда.

Керекті нәтижені алу үшін біз бөлек есептейміз: q(x) және г (ж)-ні, ал содан барып қосымша бір көбейту мен екі қосу амалын жасаймыз. Егер бұл арқылы 6 мәні дұрыс таңдалынса, онда q және г екі көпмүшесі де унимодальді болады. Біздің көріп тұрғанымыздай, оның тізбекті қабылдануы есептеу уақытын үнемдейді. Бұл аталған әдістің артықшылығы ерекше, кейде көпмүшеге қосылғыш қосамыз, содан кейін қосылған мәнді соңғы нәтижеден алып тастаймыз. Басқа сөзбен айтсақ, егер біз отыз дәрежелі көпмүшемен жұмыс жасасақ, онда оған ж31-ді қосу керек, ыдыратуды іздеу керек, содан кейін ж31-ді әрбір есептеу нәтижесінен алып тастау керек. Алгоритм бәрі бір қалған әдістерге қарағанда тез жұмыс жасайды.

Есептелетін алгоритмдер

Көпмүшелер туралы сөз айтқанның орнына мысалға назар аударайық: Көпмүшені қарастырайық:

р(х) = х7 + 4х6 - 8ж4 + 6ж3 + 9ж2 + 2х - 3.

Бірінші xj көбейткішін анықтаймыз; + Ъ (4.3) теңдігіндегі. р көпмүшесінің дәрежесі тең болады: 7, яғни. 23 — 1, сондықтан k = 3. Бұдан шығатыны:

j = 22 = 4. мәнін Ъ q және г екі көпмүше унимодальді болатындай бейнеде белгілейміз. Ол үшін p көпмүшесіндегі Oj-i коэффициентіне қарау керек және Ъ — Oj\_i — 1 қою керек. Біздің жағдайымызда бұл: Ъ = аз — 1 = 5. Біз енді келесі теңдікті қанағаттандыратын q және г мәнін іздеуіміз керек:

х7 + 4ж6 - 8х4 + 6х3 + 9х2 + 2х - 3 = (х4 + 5)<?(ж) + г(ж).

Көпмүше </ және г сәйкесінше р-ні ж4 + 5-ке бөлгендегі жеке және қалдыққа сәйкес келеді.

Қалдықпен бөлу кезінде:

р(ж) = (ж4 + 5)(ж3 + 4ж2 + Ох + 8) + (х3 - 11ж2 + 2ж - 37).

Келесі қадамда біз сол процедураны q және г көпмүшелеріне қолдана аламыз:

q(x) = (х2 - 1)(х + 4) + (х + 12), г(х) = (х2 + 1)(х - 11) + (х - 26).

Нәтижесінде аламыз:

р(х) = (х4 + 5)((ж2 - 1)(гс + 4) + (х + 12)) + ((х2 + 1)(ж - 11) + (х - 26)).

Бұл көпмүшеге қарай отырып, біздің көретініміз: ж2-ні есептеу үшін бір көбейту талап етіледі; тағы бір қосу келесі есеп үшін қолданылады: 4 = х2-х2. Бұл екі көбейту амалдарынан басқа теңдіктің оң жағын есептегенде тағы үш көбейту амалы қажет.

Бұдан басқа, 10 қосу операциясы орындалады.

Бұл әдістің салыстырмалы түрдегі нәтижелері бойынша үнемдеу айтарлықтай емес, бірақ ол жеке жағдайларға ғана айтылады.

Процедураны нақты үйрене отырып көбейтудің жалпы формуласын енгізуге болады. Байқайтынымыз: (4.3) теңдігінде бір көбейту және екі қосу амалы қатысады. Сондықтан көбейтетін сан үшін М = M(k) және қосу үшін: Л = A(k)

Біз келесідегідей рекурренттік сәйкестікті ала аламыз: М(1) = О А(1) = О

M(k) = 2M(k - 1) + 1 при k > 1 A(k) = 2A(k - 1) + 2 при k > 1.

Бұл арақатынасты шеше отырып, біз көбейту санының N/2-ге тең екенін, ал қосу саны шамамен (3N —1)/2-ге тең екенін бекіте аламыз.

Ескерілмеген көбейту амалы қалды, ол амал тізбектелген ж2, ж4, х8,..., ж2 мәндерін есептеу үшін қажет;

k — 1 көбейтіледі. Сондықтан жалпы көбейту саны шамамен тең болады: N/2 + Iog2 N.

Болжама өңдеу. N дәрежелі көпмүшенің мәнін есептейтін операциялардың саны, онда стандартты алгоритмдердің, Горнер схемасының және болжамды коэффициенттерді өңдеудің салыстырмалы анализдері енгізілген. Соңғы екі алгоритмдерді салыстыру арқылы біз N/2 — Iog2 N көбейту амалын үнемдедік, ал қосымша тағы (N — 1)/2 қосу операциясы. Көптеген есептеулерде көбейтуді қосуға алмастыру тиімді болып есептеледі, сондықтан болжамды коэффициенттік өңдеу тиімділігін арттырады.

**Регистрлер саны шектеулі машиналар**

Регистрлер саны шектеулі машиналар [Ве­ре­ща­гин,Шень,1999,Б.129]. деп аталатын келесі есептеу моделін қарастырайық.

Мұндай ма­ши­ныаның про­грам­масы мәндері на­ту­ра­л сандар болатын айнымалылырдың шектеулі санын қолданады. Сандардың размері кез келген болуы мүмкін, ма­ши­наның шексіз көлемді жады болады.

*Про­грам­ма* реті бойынша нөмірленген *ко­мандалардан тұрады*. Әрбір ко­ман­да келесі түрдің бірі болуы мүмкін [Ве­ре­ща­гин,Шень,1999,с.129]:

(1) a:=0 (нөлдеу, тазалау ко­ман­дасы)

(2) a:=b (көшіру ко­ман­дасы ко­пи­ро­ва­ния)

(3) a:=b+1 (1-ді қосу ко­ман­дасы)

(4) a:=b-1 (1-ді азайту ко­ман­дасы)

(5) goto <Но­мер> (шартсыз өту ко­ман­дасы)

(6) if a=0 (шертты өту ко­ман­дасы) thengoto <Но­мер 1> else goto <Но­мер 2>

(7) stop (тоқтату ко­ман­дасы)

goto  ко­ман­дасы әрқашан көрсетілген нөмерлі командаға өтуді іске асырады. Айнымалылардың мәндері натурал сандар 0-1 айырмасы 0-ге тең деп санаймыз (немесе *ава­ри­я*).

stop ко­ман­дасына жеткенде программа жұмысын тоқтатады.  Мысалы, [Ве­ре­ща­гин,Шень,1999,б.130]**.**

Екі санды қосу программасы  a  және b айнымалылыарныдағы сандардың қосындысын с айнымалысына орналастырады.

1 c:=a  
2 if b=0 then goto 6 else goto 3  
3 c:=c+1  
4 b:=b-1  
5 goto 2  
6 stop

Азайту, көбейту, бөлу (азайтумен жасалады), дәрежеге алу, жай сан екенін тексеру, n-ші жайсанды іздеу сияқты есептерді де осылай жеңіл программасын құруға болады.

Тью­рин­г ма­ши­насына қарағанда бұл тіл үйреншікті, барық алгоримдерді оңай программалауға болады. Тек бұл машинада массивтер жоқ, алайда кез келген размерлі натурал сандар болғандықтан және операциялар саны маңызды емес болғандықтан, биттердің массивінің орнына сандарды қолдануға болады, мысалы: (a,b,c,d,e) мына сан 2a3b5c7d11e түрінде көрсетуге болады. a[i]:=b және b:=a[i] опе­ра­циялары  a,b,i  және бірнеше айнымалылары бар шағын про­грам­маламен алмастырылады.

**Бұл модельдегі есептелетін функция.**

Про­грам­мада x , y  сияқты екі айнымалы бар болсын, x-ке   n санын, ал қалған айнымалыларға ноль берейік. Про­грам­маны рорындатамыз, егер ол тоқтамаса, онда n нүктесінде есептелетін функцияның анықталмағаны. Егер программа жұмысын тоқтатса, онда y айнымалысының мәні программада n нүктесінде есептелетін функцияның мәні болады.

**Анықтама**[Ве­ре­ща­гин,Шень,1999,б.131]**.**

Фун­кция *есептелетін* (*осы мо­де­лде*) деп атадады, егер оны есептейтін про­грам­ма бар болса.Есепті талдағанда оны шешудің полиномды алгоритмі бар-жоғын білу керек. Бұл сұраққа NP-толықтық теориясы жартылай жауап береді.

Бұл алгоритмнің қиындығын анықтау:

*n* – кірістің ұзындығы.

*A*1 алгоритмі *P есебін*  *O*(*n5*) қиындықпен шешеді.

*A*2 алгоритмі *P есебін*  *O*(2*n*) қиындықпен шешеді.

ЭВМ - 1 млн. опер./сек.

Онда *n =* 60 болғанда *A*1 алгоритмі **13 минут**, ал *A*2 алгоритмі –**300 жүзжылдықтан** астам уақыт жұмыс істейді. Олай болса, полиномиальды алгоритмдер – тиімді! Ал экспоненциальды алгоритмдер – тиімсіз!

**ҚОРЫТЫНДЫ:**

Параллель жүйелердегі тиімді программаларды жасау 3 негізгі компоненттен тұрады: параллель алгоритмдер, параллелдікті іске асыру құралдары, жөндеу (отладка) жүйесі. Параллелдікті іске асыру құралдары деп параллел программалардың инфрақұрылымын құрайтын программалау тілдерін немесе кластар кітапханасын түсінеді.Жай тізбекті программаларға қарағанда параллель алгоритмді жөндеу (отладка) қиын процесс болғандықтан, мұндай жүйелерде жөндеу және профильдеу жүйелері маңызды бөлік болып табылады. Параллель алгоритмдер тағы бір ресурсты ескеруді талап етеді: түрлі процессорлар арасындағы байланыстардың ішкі жүйелері.Параллель алгоритмдер көппроцессорлы жүйелерді дамыту үшін, өазіргі процессорлардағы ядролар санын арттыру үшін аса маңызды. Экспоненциальды күрделілік немесе  экспоненциальды уақыт — алгоритмдер күрделілігі [теорииясында е есепті шешуге кеткен уақыт,  ол есептің өлшеміне байланысты [экспонентамен](http://ru.wikipedia.org/wiki/%D0%AD%D0%BA%D1%81%D0%BF%D0%BE%D0%BD%D0%B5%D0%BD%D1%82%D0%B0) шектелген. Басқаша айтсақ,](http://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BE%D1%80%D0%B8%D1%8F_%D1%81%D0%BB%D0%BE%D0%B6%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D0%B9) егер есептің өлшемі сызықты артса, оны шешу уақыты экспоненциальды түрде артады. Регистрлер саны шектеулі машиналар

про­грам­масы мәндері на­ту­ра­л сандар болатын айнымалылырдың шектеулі санын қолданады. Сандардың размері кез келген болуы мүмкін, ма­ши­наның шексіз көлемді жады болады.

Қолданылған әдебиеттердің тізімі: [1]-[24]