**Лекция № 1.** Алгоритмдерді формализациялау ұғымы. Алгоритмдер классификациясы. Негізгі есептеу моделдері. «Алгоритм» ұғымының математикалық модельдерінің эквиваленттілігі, алгоритмдер теориясының тезистері.

**МАҚСАТЫ** **:** Алгоритмдер теориясы ,алгоритмдерді формализациялау ұғымы. алгоритмдер классификациясы, алгоритмдерді құру және орындату, есептеу моделдері ұғымдарымен таныстыру.

**СҰРАҚТАР**:

1. Алгоритмдер теориясы
2. Алгоритмдерді формализациялау ұғымы.
3. Алгоритмдер классификациясы.
4. Негізгі есептеу моделдері.
5. «Алгоритм» ұғымының математикалық модельдерінің эквиваленттілігі
6. Алгоритмдер теориясының тезистері.

**НЕГІЗГІҰҒЫМДАР:** Алгоритмнің классикалық теориясы . Алгоритмдерді асимтотикалық талдау теориясы . Есептеу алгоритмдерін практикалық талдау теориясы. есептеу моделдері. «Алгоритм» ұғымының математикалық модельдерінің эквиваленттілігі, алгоритмдер теориясының тезистері.

Алгоритммен келесі зерттеу салалары байланысты:

***Алгоритмді талдау.*** Бұл саланың зерттеу пәніне алгоритм жұмысының сипаттамаларын анықтау жатады. Мысалы: әдетте алгоритмнің жылдам орындалуы талап етіледі (метод оптимизация)

***Алгоритмдер теориясы.*** Бұл салада берілген шамаларды есептеудің тиімді алгоритмдердің болуы/болмау мәселелері қарастырылады.

***Алгоритмдерді құру.*** Бұл сала алгоритмдерді жазуда қолданылатын стандартты әдістер мен тәсілдерді қарастырады. Мыс: Блок-схема.

**Алгоритмдер теориясы**алгоритмдердің жалпы қасиеттері мен заңдылықтарын, оларды көрсетудің түрлі формальды модельдерін зерттейтін ғылым саласы. Алгоритм ұғымын формальдау арқылы алгоритмдерді тиімділігі бойынша салыстыруға, эквиваленттілігін тексеруге, қолдану салаларын анықтауға болады.

**Алгоритмдер теориясының міндеттеріне** есептің алгоритмдік шешімі болмауын формальды түрде дәлелдеу, алгоритм күрделілігін асимптотикалық талдау, күрделілік кластарына сәйкес алгоритмдерді жіктеу, алгоритмдер сапасын салыстырмалы түрде бағалаудың өлшемдерін жасау және т.б. жатады.

1930-жылдары жасалған алгоритмдердің формальды модельдері (Пост, Тьюринг, Черч), 1950- жылдары жасалған Колмогоров пен Марковтың модельдері тең , өйткені олар бір біріне мына мағынада эквивалентті: бір модельде шешімі табылған проблемалардың кез келген класы, екінші модельде де шешімі болады.

Қазіргі кезде алгоритмдер теориясының негізінде алынған практикалық ұсыныстар программалаық жүйелерді жасау мен жобалау саласында кеңінен қолданылуда.

Алгоритмдер теориясының шығу тарихы:

Алгоритм ұғымы мен аксиомалық жүйе антика заманынан басталады (Евклид). Алайда алгоритмнің нақты математикалық анықтамасы әлі болған жоқ . XXғ басы Гильберт және оның мектебі арқылы осы ұғымды формализациялады. Кейнс пен оны Гедель жалғастырды және математикалық тұрғыдан кез-келген формуланы автоматты тексеруді құрды (1931).

Алгоритмдер теориясының дамуы алғаш формальды жүйелердің толықсыздығы туралы теораманы 1931 жылы К. Гёдельдің дәлелдеуінен басталды. Осы теоремаға қатысты көптеген математикалық проблемалардың алгоритмдік шешімі болмауы туралы жорамал алгоритм ұғымының стандартталуын қажет етті.

Кейін алгоритм ұғымын стандарттау Тьринг, Черч, Пост т.б. жұмыстарында жалғастырылды. Бұлардың машиналары эквивалентті болды. Алгоритмнің сәтті стандарталған вариантын А.Марков «нормальді алгоритм» ұғымын енгізу арқылы жасады. Гёдель еңбектеріне негізделе отырып [С. Клини](http://ru.wikipedia.org/wiki/%C3%90%C2%9A%C3%90) алдыңғы аталғандарға эквивалентті [рекурсивті функци](http://ru.wikipedia.org/wiki/%C3%90%C2%A0%C3%90%C2%B5%C3%90%C2%BA%C3%91%C2%83%C3%91%C2%80%C3%91%C2%81%C3%90%C2%B8%C3%90%C2%B2%C3%90%C2%BD%C3%90%C2%B0%C3%91%C2%8F_%C3%91%C2%84%C3%91%C2%83%C3%90%C2%BD%C3%90%C2%BA%C3%91%C2%86%C3%90%C2%B8%C3%91%C2%8F)я ұғымын енгізді.Алгоритмдер теориясына Д.Кнут, А.Ахо, Дж. Ульман да өз еңбектерін сіңірді.

**Қазіргі кезде алгоритмдер теориясы негізі 3 бағытта дамып келеді:**

*Алгоритмнің классикалық теориясы* **–** есептерді формальды тілдер терминдерімен беру (формулировка задач) проблемасын зерттеді. Есептерді күрделілік кластары бойынша (P,NP, т.б) классификациясын жасайды, есептердің шешімін табу ұғымын енгізді.

*Алгоритмдерді асимтотикалық талдау теориясы*. Алгоритмнің , соның ішінде рекурсивті алгоритмдердің, орындалуының уақытының немесе ресурстарының асимтотикалық бағасын алудың тәсілін қарастырады. Асимтотикалық талдау енгізілетін деректер көлемінің өсуіне байланысты алгоритмдердің ресурстарға (мыс: орындалу уақыты) қажеттілігін бағалауға мүмкіндік береді.

*Есептеу алгоритмдерін практикалық талдау теориясы.* Тиімді алгоритмдерді таңдау әдістемесін жасау, алгоритмдердің сапасын тексерудің практикалық өлшемдерін іздеу, функцияларды интервалдық талдау, күрделіліктің нақты функцияларын алу және т.б. сияқтарды есептерді шешумен шұғылданады.

**Қазіргі заманғы алгоритмдер теориясының даму бағыттары мен шешетін негізгі міндеттері:**

«Алгоритм» ұғымын формальдау және формальды алгоритмдік жүйелерді (есептеулер модельдерін) зерттеу;

Есептің алгоритмдік шешімі болмайтынын дәлелдеу;

Алгоритмдердің дұрыстығы мен эквиваленттілігін формальды түрде дәлелдеу;

Есептерді классификациялау, күрделілігі бар кластарды анықтау және зерттеу;

Есептің күрделілігінің теориялық төменгі бағасын дәлелдеу;

Тиімді алгоритмдерді құру әдістерін табу;

Итерациялық алгоритмдердің күрделілігін асимптотикалық талдау;

Рекурсивті алгоритмдерді зерттеу және талдау;

Алгоритмдердің қиындығының нақты функцияларын табу;

Алгоритмдердің классификациясын жасау;

Есептер мен алгоримдердің сыйымдылығы бойынша күрделілігін (жад ресурстары бойынша) зерттеу;

Алгоритмдердің ресурстық тиімділігін салыстырмалы түрде бағалаудың критерийлерін және оларды салыстырмалы талдаудың әдістерін жасау.

Алгоритмдер теориясында алынған нәтижелер екі аспектіде: теориялық және практикалық аспектілерде қолданылады.

**Теориялық** **аспект:** Есептің алгоритмдік шешімі болатындығын немесе оны шешудің нақты алгоритмі болмайтындығына есепті зерттеу нәтижесінде алгоритмдер теориясының жауап беру мүмкіндігі **теориялық** **аспектіге** жатады.

Алгоритмдік шешімі болмайтын есептер Тьюринг машинасын тоқтату есебіне келтіріледі. Ал алгоритмдік шешімі болатын есептер үшін олардың NP-толық есептер класына тиесілі ме екендігі анықталады. Егер тиесілі болса, онда бастапқы деректері үлкен болатын есептердің нақты шешімін алу үшін қанша уақыт кететінін анықтауға немесе оны шешудің жылдам нақты алгоритмі болмайтындығын айтуға мүмкіндік береді.

**Практикалық аспект:** алгоритмдер теориясының, әсіресе асимптотикалық және практикалық талдаудың әдістері мен әдістемелері келесі мүмкіндіктерді береді:

Берілген есепті шешуге арналған алгоритмдер жиынынан жасалатын программалық жүйедегі олардың ерекшелігін ескеретін тиімді алгоритмді таңдауға мүмкіндік береді.

Күрделілік функциясы арқылы күрделі есептерді шешудің уақыттық бағалауларын алады. Белгілі уақыт ішінде қандай да бір есептің шешуі болмайтындығына шынайы баға беріледі. Ақпаратты өңдеу саласындаың маңызды деген есептерін шешудің тиімді алгоримтмдерін құру мен жетілдіру.

Мысалы: алгоритмді жүзеге асыратын программаның орындалу уақытына немесе пайдаланатын минималды жад көлеміне шектеулер қойылғанда түрлі алгоритмдерден таңдау жасалынады; қиындық функциясы арқылы күрделі есептерді шешу уақыты анықталады; берілген уақыт ішінде есепті шешу мүмкін болмайтындығы шынайы бағаланады. Бұл қазіргі кезде криптографиялық әдістер үшін, ақпаратты өңдеу саласында практикалық жағынан маңызды есептерді шешудің тиімді алгоритмдерін жасау мен жетілдіру үшін аса сұранысқа ие болып отыр.

**Алгоритм –­­­** өңдеудің қарапайым орындалатын тактілерін қолдануға негізделген қандай да бір әдістің нақты және шекті (ақырлы) сипаттамасы. **Алгоритм** **­­­­–** мүмкін болатын бастапқы деректер класына ортақ есептің шешімін табуға арналған нақты анықталған және орындалатын қарапайым операциялардың ақырлы тізбегінен тұратын, қандай да бір тілде жазылған ақырлы нұсқаулар.

«Алгоритм» ұғымының бірыңғай анықтамасы жоқ. Көптеген анықтамалардың ішінде неғұрлым сәтті берілгені орыс ғалымдары А.Н.Колмогоров пен А.А.Марковтың анықтамалары:

**Анықтама 1. (Колмогоров):** «**Алгоритм** **–** нақты ережелермен қатаң түрде орындалатын, қандай да бір қадамдардан соң қойылған есептің шешіміне әкелетін есептеулердің кез келген жүйесі»

**Анықтама 2. (Марков):** **Алгоритм** **–** өзгертілетін бастапқы деректерден ізделінді нәтижеге әкелетін есептеу процесін анықтайтын нақты ереже.

Бұл анықтамалардан алгоритмге қойылатын келесі ортақ талаптарды атап өтейік:

алгоритм қарапайым орындалатын саны шектеулі ережелерден тұруы тиіс, яғни жазбалардың шектеулі болуы талабын қанағаттандыруы тиіс;

алгоритм шектеулі қадамнан соң есепті шешуі тиіс, яғни әрекеттердің шектеулі болу талабын қанағаттандыруы тиіс;

алгоритм барлық мүмкін болатын бастапқы деректер үшін біреу ғана болуы керек, яғни әмбебаптық талабын қанағаттандыруы тиіс;

алгоритм қойылған есептің дұрыс шешімін табуы керек, яғни дұрыс болу талабын қанағаттандыруы тиіс.

Бұл анықтамаларда әрекетті орындаушы көрсетілуі тиіс және ол орындайтын «қарапайым» операцияларға не жататыны нақтылануы тиіс. Алгоритмді әртүрлі формада бейнелеуге , яғни беруге болады. Кейде әрекеттер ретін графикалық түрде көрсету түсінікті болады, ал оны сөзбен сипаттау қиынға түседі. Сондықтан **алгоритмнің нақты формальды анықтамасы** туралы ұғым арнайы математикалық конструкцияларды **–** формальды алгоритмдік жүйелер немесе Пост машинасы, Тьюринг машинасы, Черчтің рекурсивті-есептелінетін функциялары сияқты есептеулер модельдерін енгізумен, бұл формализмнің эквиваленттілігі туралы тезистің айтылуымен және «алгоритм» ұғымымен тікелей байланысты болды.

Z есебінің бастапқы деректерінің жиыны — DZ, aл R — мүмкін болатын нәтижелер жиыны болсын, онда алгоритм DZ → R бейнелеуін іске асырады деп айтады. Егер алгоритм тек кейбір d∈DZ үшін ғана дұрыс нәтиже берсе, онда алгоритм **жеке жағдайдың алгоритмі** деп аталады. Егер алгоритм барлық d∈DZ үшін дұрыс нәтиже берсе, онда алгоритм **толық алгоритм** деп аталады.

Қойылған есептің шешімін арифметикалық амалдарды орындауға келтіретін алгоритмдер сандық алгоритмдер деп аталады.

Д.Кнут «Искусство программирования. Основные алгоритмы» атты кітапта **алгоритмнің бес маңызды қасиетін** көрсеткен:

1) *Ақырлы болуы* **–** алгоритмнің ақырлы қадамынан соң аяқталуы тиіс;

2) *Анықтылығы* **–** алгоритмнің әр бір қадамы нақты анықталуы керек.

3) *Енгізілетін деректердің болуы* **–** алгоритм жұмысының басына дейін берілетін кірістік деректер болады немесе олар орындалу барысында динамикалық түрде анықталуы мүмкін

4) *Шығыстық деректердің болуы* **–** алгоритмде кірістік деректермен белгілі бір байланысы бар немесе бірнеше деректер болады

5) Тиімділік

**Қиындық фунциялары негізінде алгоритмдер классификациясы**

Практикалық есептерді шешуде алгоритмдерді қолдану кезінде біз есепті шешудің алгоритмін рационалды таңдау мәселесіне кезігеміз. Таңдау мәселесінің шешімі өз кезегінде алгоритмнің формалды моделіне сүйенетін салыстырмалы бағалар жүйесін құрумен байланысты.

Келешекте ортақ мәселеге қолданылатын Посттың анықтамаларын ұстана отырып дұрыс және финитті алгоритмдерді, яғни ортақ мәселенің бір шешімін беретін алгоритмдерді қарастырамыз. Формальды жүйе ретінде фон Неман архитектуралы процессорын, адрестік жады мен жоғары деңгейлі тілмен сәйкестендірілген «элементар» операциялар жиынтығын қамтитын абстрактты машинаны қарастырамыз.

ҚИЫНДЫҚ ФУНКЦИЯСЫ ТҮРІНЕ БАЙЛАНЫСТЫ АЛГОРИТМДЕР КЛАССИФИКАЦИЯСЫ

Бастапқы деректердің алгоритмнің қиындығына әсеріне байланысты, алгоритмдерді анализ жасауда практикалық мәні бар келесі классификациялар ұсынылуы мүмкін:

1. Қиындығына байланысты сандық-тәуелді алгоритмдер

Бұл қиындығы нақты кірістің өлшемінен ғана тәуелді, және нақты мәндерден тәуелсіз алгоритмдер:

(D) = (|D|) = (N)

Сандық-тәуелді қиындығы функцияларына мысал ретінде массив және матрицалармен жұмыс жасайтын операциялар – матрицаларды көбейту, матрицаны векторға көбейту, жіне т.б. орындайтын стандартты функциялардың алгоритмдерін алуға болады.

2. Қиындығы бойынша параметрлі-тәуелді алгоритмдер

Бұл қиындығы кірістің өлшемімен емес(шарт бойынша, бұл топтың кіріс өлшемі тұрақты), жадының өңдейтін сөздерінің нақты мәндерімен анықталатын алгоритмдер:

(D) = (,…,) = (,…,), m =< n

Параметрлі-тәуелді қиындығы функцияларына мысал ретінде берілген дәлдікпенсәйкес дірежелік қатарды есептеу арқылы есептейтін стандартты функцияларының алгоритмдерін алуға болады. Бұдан көретініміз, мұндай алгоритмдер, кірісте екі сандық мәнді - функция аргументі және дәлдікті ала отырып, мәннен өте тәуелді операциялар санын орындайды.

а) Тізбекті көбейту арқылы  есептеу (x, k) = (k).

б) = (/n!) –ті есептеу, = (x, ) дәлдігімен

3. Қиындығы бойынша сандық-параметрлік алгоритмдер

Бірақ, көптеген тәжірибелік жағдайда қиындығы функциясы кірістегі деректер санынан да, кіріс деректерінің мәндерінен де тәуелді:

(D) = (||D||, ,…,) = (N, ,…, )

Мысал ретінде парамерлі-тәуелді циклі дәлдік бойынша сандық-тәуелді өлшем бойынша фрагментін қамтитын сандық тәсілдердің алгоритмін алуға болады.

Қиындығы бойынша реттік-тәуелді алгоритмдер

Көптеген параметрлітәуелді алгоритмдердің арасынан тағы бір топты белгілейміз. Олар үшін операциялар саны бастапқы обьектілердің орналасу ретінен тәуелді.

D жиынын (,…,), и ||D||=N элементтерінен тұрсын,

 = {( ,…,)}- ,…, ішіндегі барлық реттелген Nдер жиыны, мұндағы ||=n!.

Егер (i) (j), мұндағы i, j є , онда алгоритмді ретті-тәуелді деп атаймыз.

Бұларға мысал ретінде сұрыптау алгоритмдер, іздеу алгоритмі, массивте максимум және минимумды іздеу алгоритмдерін алуға болады.

n элементтен тұратын S массивінде максимумды іздеу алгоритмін толығырақ қарастырайық:

|  |  |
| --- | --- |
|  | (орындалған теңестіру операцияларының саны массивте орналасу ретінен тәуелді) |

**Алгоритмдер классификациясы.** Күрделілілік тұрғысынан алгоритмдердің екі үлкен класы бар − қайталауы бар алгоритмдер және рекурсивті алгоритмдер.Бір есепті көптеген алгоритммен шешуге болады. Олардың әрқайсысының жұмысының тиімділігі әртүрлі сипаттаулармен сипатталады. Алгоритмді талдамас бұрын ең алдымен берілген алгоритм есепті дұрыс шешетінін дәлелдеуіміз керек. Егер есепті дұрыс шешпесе тиімділік мәселесінің мәні жоқ. Егер алгоритм қойылған есепті шешсе, оның қаншалықты тиімді екенін қарастыруымыз керек. Сондықтан оның тиімділігінін анықтау үшін алгоритмдерді талдауымыз қажет.

Қайталау алгоритмінің негізінде шартты өрнектер мен цикл жатыр; мұндай алгоритмдерді талдауда циклдын ішінде қолданылатын операция саны мен цикл итерациясының санын бағалау керек. Рекурсивтік алгоритмдер үлкен есепті фрагменттерге бөледі және әрбір фрагменттерді жеке қолдануға мүмкінлік береді. Осындай алгоритмдерді кейде «бөліп алда біле» алгоритмі деп айтайды және оларды қолдану өте тиімді болуы мүмкін. Үлкен есептердікішіге бөлу жолымен шешу үрдісінде үлкен емес, қарапайым және түсінікті алгоритм құрылады. Рекурсивті алгоритмді талдауда есепті бөлімге бөлуге қажетті операцияның санын санау, ірбңр бөлімдегі алгоритмнің орындалуы және есепті толық шешудегі әрбір бөлімнің нәтижесін біріктіру керек. Осы ақпараттарды және неше бөлік екенінің саны, олардың өлшемінен алгоритмнің күрделілігінің рекуррентік қатынасын шығарамыз.

Алгоритмдерді әртүрлі белгілер бойынша жіктеуге болады. Мысалы, жалпы [комбинаторлық](http://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%82%D0%BE%D1%80%D0%B8%D0%BA%D0%B0) алгоритмдер, тағы алгоритмдер, [максимальды ағынды іздеу алгоритмдері](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), [максимальды жұп сәйкестігін іздеу алгоритмдері](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), [іздеу алгоритмдері](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), [Алгоритмы на строках](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC_%D0%BD%D0%B0_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0%D1%85&action=edit&redlink=1), жолды [іздеу алгоритмдері](http://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D0%BA%D1%81%D0%B8%D0%BC%D0%B0%D0%BB%D1%8C%D0%BD%D1%8B%D0%B9_%D0%BF%D0%BE%D1%82%D0%BE%D0%BA), [Бұтақтармен](http://ru.wikipedia.org/w/index.php?title=%D0%90%D0%BB%D0%B3%D0%BE%D1%80%D0%B8%D1%82%D0%BC%D1%8B_%D0%B2%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B5%D0%BD%D0%B8%D1%8F_%D1%80%D0%B0%D1%81%D1%81%D1%82%D0%BE%D1%8F%D0%BD%D0%B8%D1%8F_%D0%BC%D0%B5%D0%B6%D0%B4%D1%83_%D1%81%D1%82%D1%80%D0%BE%D0%BA%D0%B0%D0%BC%D0%B8&action=edit&redlink=1) жұмыс алгоритмдері, сұрыптау алгоритмдері, сығу алгоритмдері, үлестірілген жүйелер алгоритмдері, операциялық жүйелердегі алгоритмдер, тиімділеу алгоритмдері және т.б.

Есептеу моделіне байланысты сандық, логикалық, продукциялық, нейрожелілік, параллель, полиномдық, құмырсқа, генетикалық және т.б. бөлінеді.

Есептеулер модельдерінің үш түрі бар:

1) Тьюринг машинасы

2) Лямбда есептеу (рекурсия)

3) Комбинаторлық логика

**ҚОРЫТЫНДЫ:**

Алгоритмді талдау саласында саланың зерттеу пәніне алгоритм жұмысының сипаттамаларын анықтау жатады. Алгоритмдер теориясы саласында берілген шамаларды есептеудің тиімді алгоритмдердің болуы/болмау мәселелері қарастырылады. Алгоритмдерді құру саласында Бұл сала алгоритмдерді жазуда қолданылатын стандартты әдістер мен тәсілдерді қарастырады.

Алгоритмдер теориясы алгоритмдердің жалпы қасиеттері мен заңдылықтарын, оларды көрсетудің түрлі формальды модельдерін зерттейтін ғылым саласы. Алгоритм ұғымын формальдау арқылы алгоритмдерді тиімділігі бойынша салыстыруға, эквиваленттілігін тексеруге, қолдану салаларын анықтауға болады.

Қолданылған әдебиеттердің тізімі: [1]-[23]