Computational Thinking

2017 2학기

10-계산의 한계

박종혁 교수 (컴퓨터공학과)

jhpark1@seoultech.ac.kr

http://www.parkjonghyuk.net

학습개요

- 10장 계산의 한계
  1. 컴퓨터에서의 성능은 어떻게 측정되는가?
  2. 물리적인 한계의 평가
  3. 벤치마크
  4. 성능 계산하기
  5. 비실용적인 알고리즘
  6. 불가능한 알고리즘
  7. 추상적인 한계
- 조별과제
  - 다음 주 강의 시간 발표

학습목표

- 문제 해결 잠재력에 있어서 컴퓨터의 고장과 한계가 존재하며 계속 그럴 것임을 이해할 수 있다.
- 계산 성능은 저장 용량과 처리 속도라는 두 가지의 문제인데 처리속도가 더 중요함을 이해할 수 있다.
- 대부분 소형화 덕택에 현재까지 무어의 법칙이 달성되었지만, 최근에는 멀티코어 프로세서들이 이에 기여를 하고 있으며, 그러한 개선이 영원히 계속될 수 없음을 이해할 수 있다.

학습목표

- 문제 해결을 위한 컴퓨터의 능력을 측정하기 위해 벤치마크가 사용될 수 있지만, 벤치마크는 일반화 되지 못할 수도 있는 단지 매우 구체적인 하나의 데이터 지표를 제공할 뿐임을 이해할 수 있다.
- 직선의 그래프로 나타낼 수 있는 비례 증가를 통해서 일반적인 선형 알고리즘들을 알아볼 수 있다.
- 이진검색은 데이터의 중간에서 반복된 선택을 통해서 이루어지며, 이 알고리즘은 단지 정렬된 데이터에 대해서만 동작하지만 선형검색보다 성능이 상당히 좋음을 인지할 수 있다.
- 일반적으로 다항식 성능을 가진 알고리즘들은 컴퓨터로 처리하기에 쉬운 것으로 인식되고 있음을 알 수 있다.

학습목표

- 지수 알고리즘들은 적은 양의 데이터를 제외하고 일반적으로 컴퓨터로 실행하기에 비실용적인 것으로 간주되고 있음을 알 수 있다.
- 멈춤 문제와 같이 컴퓨터 프로그램에 의해 결코 해결될 수 없는 문제점들이 있음을 이해할 수 있다.
- 튜링 테스트와 이것이 컴퓨터 지능과 어떻게 관련이 지어지는지에 대해 설명할 수 있다.
- CAPTCHA가 얼마나 유용한지와 이것이 일종의 튜링 테스트의 반전을 나타내고 있음을 이해할 수 있다.

❖ 무어의 법칙(Moore's law)

- 컴퓨터 하드웨어의 처리능력은 주로 18개월마다 두 배가 된다는 법칙
- 지수적 증가(exponential growth)
- 계산 장치에 대하여 무어의 법칙은 계속 될 수 있는가?

시간경과	증가된 성능
0년	1배
1.5년	2배
3년	4배
4.5년	8배
6년	16배
7.5년	32배
9년	64배
10.5년	128배
12년	256배
13.5년	512배

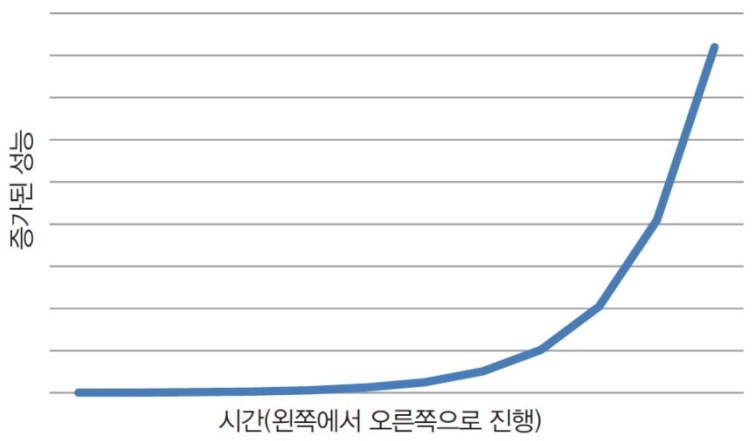


그림 10.1 무어의 법칙의 지수적 증가

❖ 성능

- (1) 처리속도를 측정, 시간(time)
- (2) 기억장치 크기를 측정 공간 (space)(메모리 공간의 양, 디스크 용량)

❖ 처리속도

- 클럭 속도(clock rate)
- 1GHz는 초당 십억 번의 순환 사이클
- 더 높은 클럭 속도를 가진 프로세서가 더 빠름

10.1 컴퓨터에서 성능은 어떻게 측정되는가?

- 더 높은 클럭 속도를 가진 한 종류의 프로세서가 더 느린 클럭 속도를 가진 다른 종류의 프로세서보다 더 빠를 수도 또는 그렇지 않을 수도 있음. 그 이유는

  1. 서로 다른 유형의 프로세서들은 각 명령어 마다 클럭 사이클의 수가 달라지기 때문

  2. 서로 다른 유형의 프로세서들은 각기 다른 종류의 명령어들을 가지기 때문

- MIPS(millions of instructions per second)
  : 초당 백만 개의 명령어
- MFLOPS(millions of floating point operations per second)
  : 초당 백만 개의 실수 연산

- 코어(core) : 종종 프로세서의 개수

  Ex) 듀오 코어(duo-core: 2중 코어) , 쿼드 코어(quad-core: 4중 코어)

- 더 적은 프로세서들을 가진 컴퓨터보다 다수의 프로세서들이 더 많은 계산을 함
- 컴퓨터의 그래픽 프로세서, 컴퓨터 메모리와 디스크 드라이브 등 은 전체 컴퓨터 속도에 영향을 끼침

  Ex) 더 빠른 프로세서는 더 느린 메모리에 때문에

  그 속도가 제한될 수 있음

❖ 무어의 법칙의 한계

- 반도체(semiconductors) 덕분에 소형화(miniaturization)가 가능해 짐
- 더 작아지고 빨라짐
- 전자의 폭보다 더 커서 전자들이 통과할 수 있는
  장치와 경로를 만들어내는 것이 점점 어려워 짐

  1. 광학 컴퓨터
  2. 생물학적 컴퓨터
  3. 양자 컴퓨터

❖ 양자 컴퓨터

- 양자이론 : 전자의 위치를 확률적으로 설명
- 더 빽빽한 데이터 형태가 될 수 있음
- 세스 로이드(Seth Lloyd)의 궁극적인 컴퓨터

	로이드의 궁극적인 컴퓨터	현재의 랩톱 컴퓨터
프로세서(연산/초)	10^{32}	10^{9}
메모리 크기	10^{16} 비트	10^{10} 비트

그림 10.2 랩톱과 비교한 로이드의 궁극적인 컴퓨터의 한계

❖ 벤치마크(benchmark)

- 응용 프로그램의 시간을 측정
- 경험적인 접근 : 직접 시간을 측정
- 속도이외의 특징들을 검사 가능
- 반응 시간 – 응용프로그램이 사용자 입력에 얼마나 빨리 반응하는가?
- 사용자 친숙도 – 응용프로그램은 사용하기 얼마나 쉬운가?
- 견고함 – 소프트웨어가 고장에 대해 얼마나 영향을 받지 않는가?
- 하드웨어와 소프트웨어 모두를 분석

10.3 벤치마크

- 각각의 컴퓨터를 벤치마크 할 때 동일한 소프트웨어와 데이터를 사용
- 일반화하기에 어려움
- 단일 환경에 대해서만 측정
- 벤치마크의 시간측정이 사실상 결과를 변경시킬 수 있음
- 반복성과 벤치마크에 영향을 주는 변수를 제어하는 것은 아직 불가능
- 최신 소프트웨어는 일반적으로 막대한 수의 잠재적인 입력을 고려하고 있고, 최신 운영체제는 사용자 제어가 별로 없거나 또는 거의 없이 다수의 응용프로그램을 동시에 실행하고 있다.
- 벤치마크에서 확신을 가지고 결론을 내릴 수 있는 거의 전부는 벤치마크가 컴퓨터의 상태가 주어진 그 시점에서 특정한 사용자의 입력이 주어졌을 때의 동작이라는 것이다.

10.4 성능 계산하기

- 성능은 데이터의 양에 따라 달라짐
- 벤치마크는 단일 데이터의 양을 고려
- 알고리즘에 포함된 작업의 수를 세는 분석적인 접근법에 의존
- 메모리로부터 데이터가 인출되거나 저장된 횟수나
  또는 실행된 컴퓨터 명령어들의 수를 셈

```
holdTicket <-- first ticket from the stack
remove holdTicket from stack
while   stack not empty AND holdTicket not winner   do
    holdTicket <-- next ticket
    remove holdTicket from stack
endwhile
if     holdTicket is winner then
    The winning ticket is holdTicket
else
    The stack contains no winning ticket
endif
```

그림 10.3 당첨 티켓을 검색하기 위한 알고리즘 A

❖ 검사되는 티켓의 개수(N)로 알고리즘의 속도를 평가

❖ 선형 알고리즘(linear algorithm)

- 정비례에 따라 수행하는 알고리즘

더미에 있는 티켓의 개수	더미에 당첨 티켓이 있는 경우	더미에 당첨 티켓이 없는 경우
0	0	0
10	5	10
20	10	20
30	15	30
N	N/2	N

그림 10.4 알고리즘 A에 대해 검사한 티켓의 평가

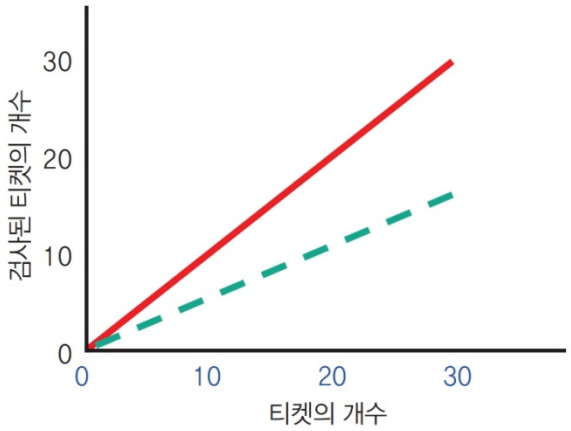


그림 10.5 알고리즘 A의 성능

```
while   the stack has more than one ticket     do
    Examine the middle ticket;
    if   middle ticket number less than winning number   than
         Remove from stack everything through middle ticket,
    else
         Remove from stack everything following middle ticket,
    endif
endwhile
if   a ticket remain in the stack and is the winner   then
    The winning ticket is the one left,
else
    The stack contains no winning ticket,
endif
```

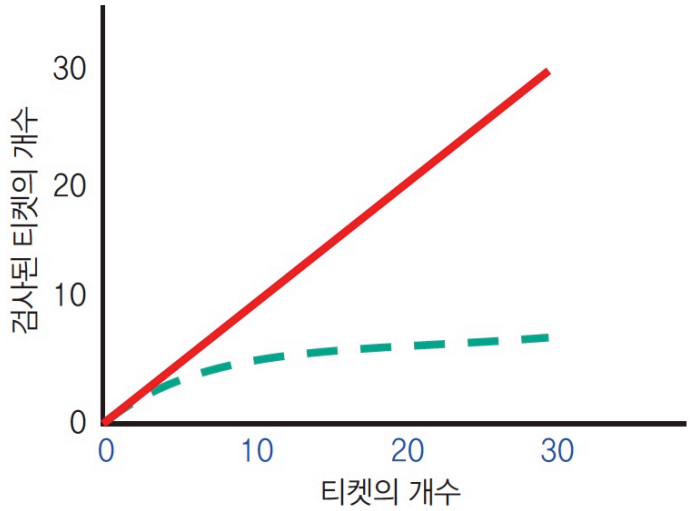
그림 10.6 당첨 복권을 찾는 이진 검색

❖ 이진 검색(binary search) 알고리즘

- 정렬이 되어 있을 때
- 단지 중간의 티켓을 검사함으로써 티켓 더미의 반절을 고려대상에서 제외
- 검사된 티켓의 개수에 대한 일반적인 공식은 log2N
- 선형 검색을 사용하는 프로그램보다 더 빠름

더미에 있는 티켓의 개수	선형 검색	이진 검색
	0	0
4	4	2
8	8	3
32	32	5
N	N	$\log_2 N$

(a)



(b)

```
holdTicket <-- first ticket from the stack
remove nextTicket from stack
while   the stack is not empty   do
    nextTicket <-- next ticket from the stack
    remove nextTicket from stack
    if   nextTicket >  holdTicket then
        holdTicket <-- nextTicket
    endif
endwhile
// holdTicket has greatest number from the stack
```

그림 10.8 복권 더미에서 가장 큰 수를 찾는 알고리즘

```
Begin with a stack that is sorted & empty.
while   the stack contains one or more tickets   do
    Find largest numbered ticket in unsorted stack.
    Move found ticket to top of sorted stack.
endwhile
```

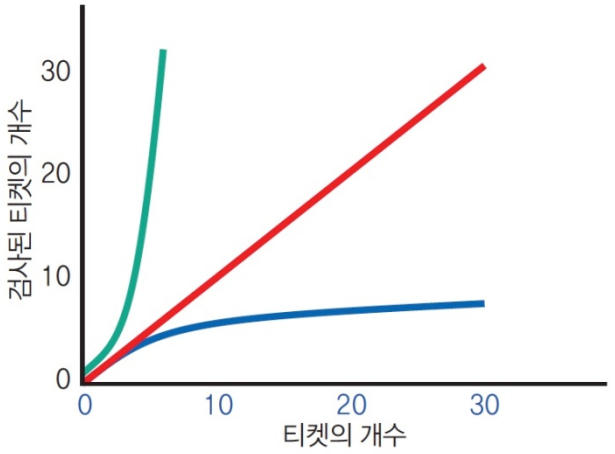
그림 10.9 복권 더미를 정렬하기 위한 알고리즘

❖ 정렬 알고리즘

- 티켓이 하나의 더미에서 또 다른 더미로 옮겨져야 하는 횟수를 셈
- 가장 큰 번호를 가진 티켓을 정렬되지 않은 더미로부터 새롭게 정렬된 더미로 옮김

검사한 총 티켓의 개수	=	가장 큰 번호를 찾기 위해 검사한 티켓의 개수	+	두 번째로 큰 번호를 찾기 위해 검사한 티켓의 개수	+	세 번째로 큰 번호를 찾기 위해 검사한 티켓의 개수	+…+	마지막으로 큰 번호를 찾기 위해 검사한 티켓의 개수
검사한 총 티켓의 개수	=	N	+	$(N-1)$	+	$(N-2)$	+…+	1
검사한 총 티켓의 개수	=	$N^2 - \frac{N}{8}$						

(a)



(b)

❖ 다항식의 성능(polynomial performance)

- 데이터 항목의 개수(N)를 세기 위한 공식이 다항식
- 프로그램들은 다항식 성능 가짐
- 최악의 경우 : m3조합(숫자 표시의 개수(m))

표시의 개수(m)	조합의 개수(m^2)	시간(조합당 5초)
40	64,000	88.89 시간
41	68,921	95.72 시간
42	74,088	102.90 시간
43	79,507	110.42 시간

그림 10.12 다이얼 표시의 개수에 따라 조합을 찾는데 필요한 조합의 개수와 시간

그림 10.11 다이얼 자물쇠

❖ 지수 알고리즘(exponential algorithm)

- 최악의 경우: 40r (다이얼 횟수 r)
- 변수(r)이 지수
- 기하급수적

다이얼 횟수(r)	조합의 개수(40^r)	시간(조합당 5초)
3	64,000	88.89 시간
4	2,560,000	296.29 일
5	102,400,000	16.22 년

그림 10.13 다이얼 횟수에 따라 조합을 찾는데 필요한 조합의 개수와 시간

❖ 다항식 증가와 지수적 증가

- 차이가 극단적
- 모든 경우를 시도하는 알고리즘들은 기하급수적이 되기 쉬움
- 지수알고리즘 대신 대안적인 알고리즘 이용

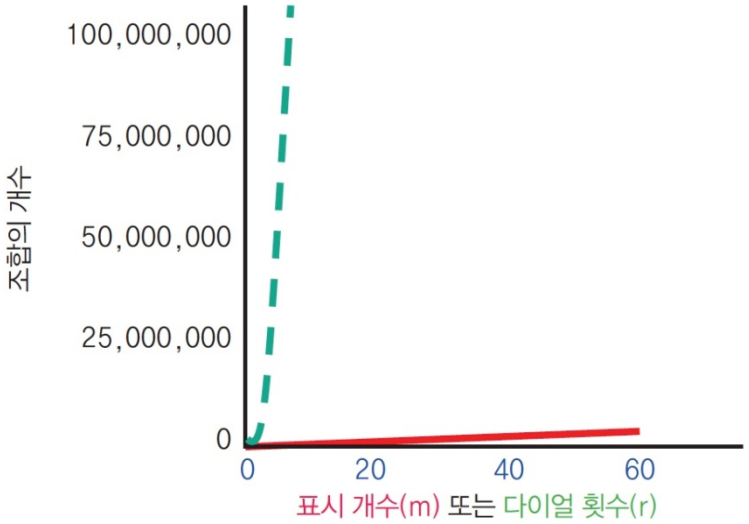


그림 10.14 자물쇠에 따른 다항식 알고리즘과 지수 알고리즘의 그래프

데이터의 양(D)	D^3	3^D
1	1초	1초
5	2.08분	4.05분
10	16.67분	16.40시간
15	56.25분	166일
20	2.22시간	110년
25	4.34시간	26,849년
30	7.50시간	6,524,296년
35	11.91시간	1,585,403,995년
40	17.78시간	385,253,170,680년*

*이 수는 현재 예측으로 우주의 나이의 거의 26배이다.

그림 10.15 D^3과 3^D 증가의 비교

❖ 멈춤 문제(halting problem)

- 무한 루프(infinite loop) : 영원히 실행되는 알고리즘

Ex)

```
while true do
        // do something
endwhile
```

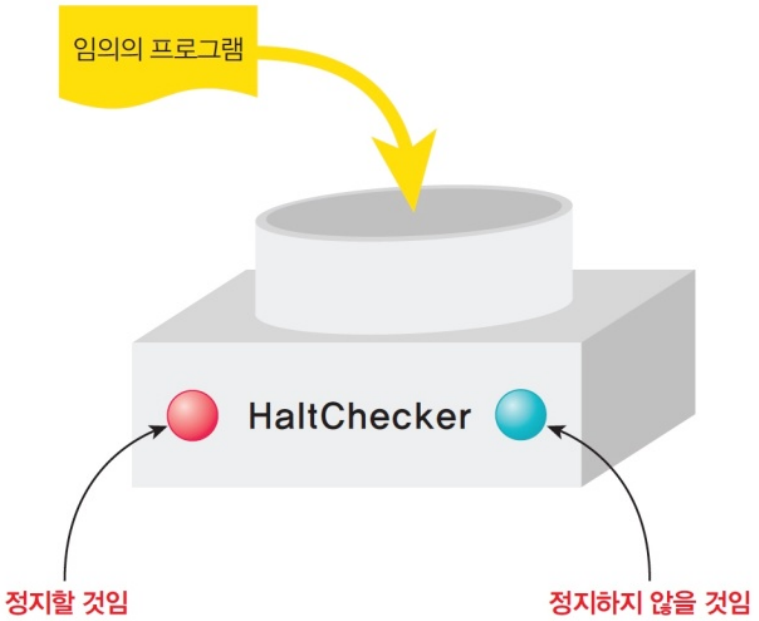


그림 10.16 HaltChecker

```
If   HaltChecker says InterstingProgram will halt    then
    while   true   do
        // do something
    endwhile
else
    // halt
endif
```

- 정지여부를 확인하는 알고리즘(HaltChecker)
- HaltChecker가 InterestingProgram은 정지할 것이라고 결정하면, InterestingProgram은 정지하지 않게 됨(올바로 동작하지 않음)
- HaltChecker가 InterestingProgram은 정지하지 않을 것이라고 결정하면, 정지(올바로 동작하지 않음)

- HaltChecker가 일부 또는 대부분의 경우에 동작하도록 하는 것은 가능하지만 항상 동작하는 HaltChecker 알고리즘을 작성하는 것은 불가능
- 어떤 프로그램도 멈춤 문제를 해결할 수 없음

❖ **계산 불가능(noncomputable - 컴퓨터로 해결 불가능)**

- 정지하는 모든 프로그램에 대하여 동작하는 HaltChecker를 작성하는 것은 가능
- 프로그램이 완료되면, HaltChecker는 그 프로그램이 정지한다는 것을 확신할 수 있다.
- InterestingProgram이 정지하지 않는 경우에, 여전히 HaltChecker와 같은 프로그램은 동작하지 않는다.

❖ 튜링 테스트(Turing test)

- 목적 : 컴퓨터 시스템이 지능이 있는지를 결정
- 질문자는 A와 B 둘 중의 어느 것이 컴퓨터이고 어느 것이 사람인지 알 수 없음
- 질문자는 문제를 물어보고 A와 B는 답변
- 질문자가 둘 중의 어느 것이 사람인지 확실하게 골라낼 수 없다면, 컴퓨터 시스템은 튜링 테스트를 통과

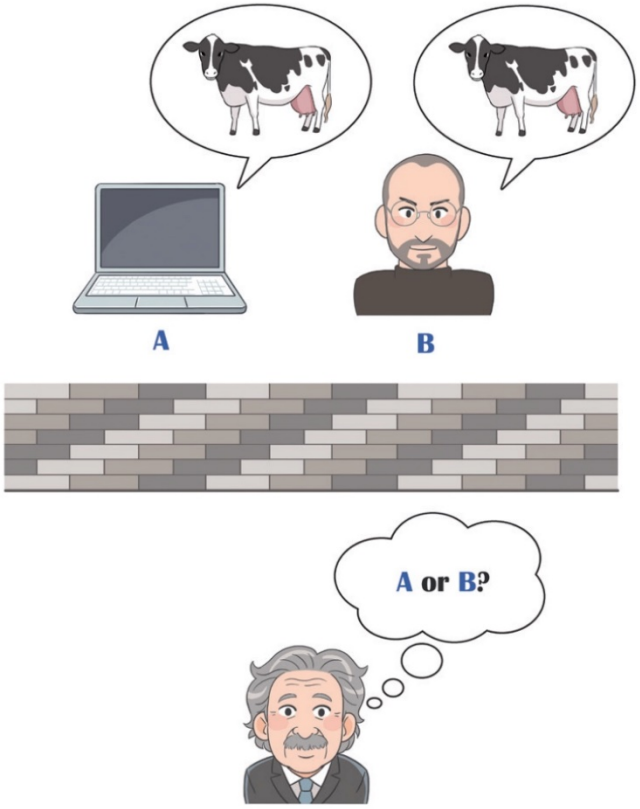


그림 10.17 튜링 테스트

❖ 한계

- 튜링 테스트가 구별하기를 원했던 것은 그림의 오렌지색 영역 뿐
- 인간 이외의 지능을 확인할 수는 없음
- 모든 인간 행동이 지능적인 것으로 특징지어질 수는 없음

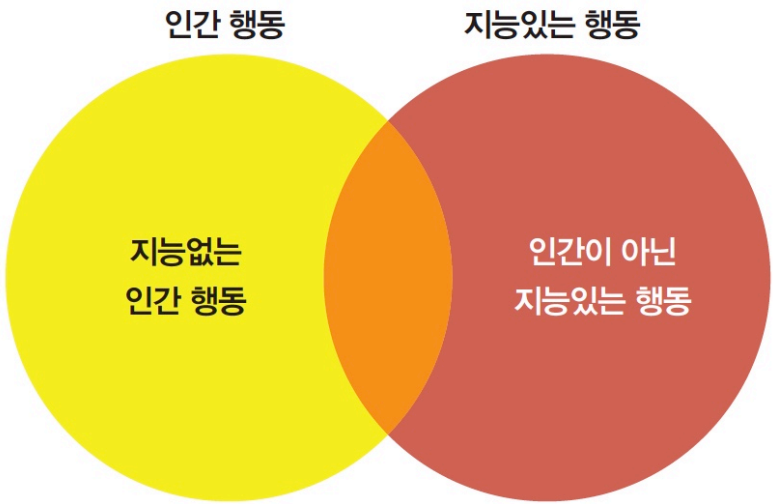


그림 10.18 인간의 행동과 지능에 대한 벤 다이어그램

❖ **캡차(CAPTCHA: Completely Automated Public Turing test to tell Computers and Human Apart)**

- 인간이 이해하는 것은 비교적 쉽지만 컴퓨터 프로그램이 이해하기에는 어렵게 만드는 방식으로 디자인된 그래픽 이미지
- 목적 : 웹사이트가 인간과 소통하고 있다는 것을 확신하기 위함
- 일종의 역 튜링 테스트
- 인간과 컴퓨터를 구별하기 위한 차이점을 이용

그림 10.19 "inherit"라는 단어에 대한 캡차(CAPTCHA)

❖ 인공 지능

- 지능 이외에 현대의 컴퓨터 시스템들이 흉내 내기 어려운 인간의 다른 특징을 흉내 내기 위한 발전

10.8 언제 이 내용을 사용할 것인가?

- 컴퓨팅 능력들의 한계
- 지수적 증가를 하는 알고리즘은 컴퓨터에서 처리하에 실용적이지 않음
- 계산할 수 없는 일부 알고리즘들이 있음
- 멈춤 문제를 가지는 알고리즘은 본래부터 불가능한 것이며 컴퓨터 능력에 어떤 제한 때문이 아님
- 지능이나 감정을 가지는 컴퓨터 시스템

10장 – 계산의 한계

- 발표방식: ppt 자료 활용 2개 조 각각 10분 발표 (질의응답 포함)

- 문제 1
  - 다음 제시된 알고리즘에 대하여 각각 가능, 불가능, 비실용적으로 구별하고 이유를 설명하여라. 또한 아래의 예시 외에 가능, 불가능, 비실용적 각각의 상황에 대한 예를 한 가지 이상 들고 이유를 설명하시오.

  1) 대한민국 인구조사 프로그램
  2) 거대한 자연수의 약수를 찾는 문제

- 문제 2
  - 튜링 테스트란 기계가 인간과 얼마나 비슷하게 대화할 수 있는지를 기준으로 기계에 지능이 있는지를 판별하고자 하는 테스트이다. 튜링 테스트를 통과한 인공지능 컴퓨터가 개발이 되고 상용화 된다면 미래의 우리의 사회는 어떻게 될 것인가?’에 대해 토의하고 발표하시오.